Universidad de Alcalá
Escuela Politécnica Superior

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

Trabajo Fin de Máster

Aplicación de Matlab/Simulink al posicionamiento y control de drones en interiores

Autor: Daniel García González
Tutor: Felipe Espinosa Zapata
Cotutor: Carlos Santos Pérez

2020

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

Trabajo Fin de Máster

Aplicación de Matlab/Simulink al posicionamiento y control de drones en interiores

Autor: Daniel García González

Tutores: Felipe Espinosa Zapata

Cotutor: Carlos Santos Pérez

TRIBUNAL:

Presidente: Marta Marrón Romera

Vocal 1º: Javier Acevedo Rodríguez

Vocal 2º: Felipe Espinosa Zapata

FECHA: Septiembre 2020

ÍNDICE

ÍNDICE DE FIGURAS

RESUMEN

El objetivo principal de este trabajo es el desarrollo de un sistema de posicionamiento en interiores de drones para el seguimiento de trayectorias. El sistema diseñado utiliza la información captada por una cámara localizada en la parte inferior del dron y se encarga de detectar e identificar marcas fijas, con posición conocida, situadas en el suelo. Se calcula la posición absoluta del dron a partir de su posición relativa con respecto a la absoluta de la marca. Para el diseño, simulación e implementación se han utilizado las herramientas proporcionadas por Matlab/Simulink, realizando ensayos reales con un minidrón Parrot Mambo.

PALABRAS CLAVE

Dron

Vehículo aéreo no tripulado

Control electrónico

Visión artificial

Tratamiento de Imagen

Estimación de estados

Filtro de Kalman

ABSTRACT

The main purpose of this project is to develop a drone positioning system for trajectory tracking applications oriented to work in indoor environments. The positioning system uses the information from the onboard downfaced camera that detects and identifies fixed marks located on the ground and whose position is known. The positioning system calculates the absolute position of the drone from the relative position of the drone with respect to absolute position of the mark. For the demonstration of the proposal, Parrot Mambo minidrone is used. Matlab/Simulink tools and software are used for the design, implementation and simulation.

KEY WORDS

Drone

Unmanned aerial vehicle (UAV)

Electronic control

Artificial vision

Image Processing

States estimation

Kalman filter

RESUMEN EXTENDIDO

Cada vez es más frecuente el uso de los drones autónomos en el entorno académico e industrial. Actualmente se cuenta con pequeños y ligeros minidrones con aplicaciones en entornos interiores como pueden ser almacenes, pabellones deportivos, centros académicos, etc.

En el presente trabajo se aborda el diseño de un sistema de posicionamiento para minidrones que se mueven de forma autónoma en entornos interiores donde no se cuenta con señal GPS.

El dron utilizado en el trabajo es el minidrón Parrot-Mambo -PM- y cuenta con los siguientes sensores:

- Unidad de medida inercial -IMU- (acelerómetro de 3 ejes y giroscopio de 3 ejes).
- Sensor de ultrasonido -sónar- situado en la parte inferior del dron y orientado hacia el suelo.
- Sensor de presión -barómetro-.
- Cámara vertical de 60 fps con resolución 160x120 px situada en la parte inferior del dron y orientada hacia el suelo.

Para el diseño, simulación e implementación de los controladores necesarios se ha utilizado las herramientas de Matlab/Simulink 2019 junto con las herramientas incluidas en el Add-on *Simulink Support Package for Parrot Minidrones* que suministra el fabricante Parrot. Estas herramientas proporcionan un modelo inicial de controlador del dron sobre el que es posible realizar modificaciones a partir de la información sensorial integrada en el dron. También se ha hecho uso de las librerías para Simulink orientadas a comunicación vía conexión bluetooth con un PC remoto. Por último, el Add-on para Simulink incluye un entorno de simulación basado en las librerías *Aerospace Blockset*, *Aerospace Toolbox* y *Simulink 3D Animation* que permite probar y extraer información del funcionamiento de los algoritmos de control y de visión artificial implementados como paso previo a la ejecución en el hardware del dron.

Las aportaciones realizadas se dividen en 3 partes principales: ajuste del controlador de posición, procesado de imagen y estimador-corrector de posición real del dron. A continuación, se describen brevemente cada una de estas partes:

- Ajuste del controlador de posición: Se ha conseguido un movimiento más amortiguado en el seguimiento de trayectorias para que la imagen obtenida de la cámara sea válida

durante mayor tiempo posible; así como reducir las oscilaciones del dron al alcanzar el punto objetivo.

- Procesado de imagen: Analizadas las restricciones de cálculo del minidrón y los problemas de incluir el canal de comunicación en el lazo de control, se ha optado por una solución embarcada basada en marcas simples monocolor ubicadas en el suelo a lo largo de la trayectoria. Estas marcas, a modo de balizas, permiten el posicionamiento absoluto del PM.

- Corrección de la posición estimada: Con la captura de la marca visual se corrige la posición estimada a partir del IMU embarcado mediante un Filtro de Kalman, con caracterización experimental de las matrices de covarianza Q y R.

Se han realizado pruebas en simulación y con el PM real, con marcas origen y destino que delimitan la trayectoria a seguir.

Las principales conclusiones obtenidas del trabajo realizado son las siguientes:

- Las ventajas del minidrón PM quedan atenuadas por su limitada capacidad de cálculo, especialmente para el procesamiento de imágenes en tiempo real. Por otra parte, la comunicación disponible permite la interacción con un centro remoto siempre que el control sea local.
- Para aprovechar la captura de la cámara integrada en el sistema de posicionamiento se requiere mejorar el sistema de estabilización original del minidrón. Para ello, se han introducido “wait points” a lo largo de la trayectoria actuando como balizas pasivas.
- El sistema de corrección implementado mediante un Filtro de Kalman, a partir de la información de la cámara, es necesario para evitar las derivas acumulativas asociadas a los sensores inerciales embarcados.
- Los modelos proporcionados por la herramienta Matlab/Simulink reducen la curva de aprendizaje de trabajo con el Parrot Mambo y son de gran ayuda en la fase de simulación de algoritmos, pero requieren de ajustes que adecúen los resultados experimentales a los previstos en simulación.

MEMORIA

1. INTRODUCCIÓN

En este apartado se describe el concepto de dron, así como su clasificación en función de forma y sistema de propulsión. También se explican algunos hitos clave en el uso de drones y se motiva el tipo de dron utilizado en el presente TFM. Por último, se describe el contenido de los capítulos que componen la memoria.

1.1. Notas sobre drones

Un dron es una aeronave que vuela sin piloto. También son conocidos como vehículo aéreo no tripulado (UAV) o sistema aéreo pilotado remotamente (RPAS). Los drones basan su estabilidad en un sistema de control que actúa directamente sobre las fuentes de propulsión en función de la información recibida de los sistemas de sensado, que principalmente lo componen una combinación de un giroscopio y un acelerómetro (IMU).

1.1.1. Clasificación de los drones o UAV

Generalmente, existen de dos tipos de drones o vehículos no tripulados:

- De ala fija: tienen diseño de avión convencional y son principalmente usados para uso militar. Se muestra un ejemplo en la Figura 1.1—1

Figura 1.1—1 - Dron de ala fija

- Multirrotores o multicóptero: Utilizan un sistema de propulsión de múltiples rotores. Existen distintas configuraciones de multirrotores en función del número de rotores como por ejemplo los tricópteros, cuadricópteros o hexacópteros, pero los más comúnmente utilizados son los cuadricópteros. En la Figura 1.1—2 se muestra un ejemplo de cada uno.

Figura 1.1—2 - De izquierda a derecha: tricóptero, cuadricóptero y hexacóptero.

1.1.2.Historia de los UAV

En sus inicios, los drones fueron desarrollados y utilizados por la industria militar. El primer uso registrado fue en 1849, cuando el ejército austriaco utilizó globos aerostáticos no tripulados en los que montaron explosivos para bombardear la ciudad de Venecia. Durante la Primera guerra mundial, fueron utilizados principalmente para tareas de vigilancia aérea o como blanco aéreo en tareas de entrenamiento de unidades antiaéreas.

El uso del radio control fue decisivo en la evolución de los UAV, permitiendo que puedan ser controlados remotamente desde tierra. Un ejemplo de esto es el “Aerial Target”, desarrollado por el ejército de Reino Unido a finales de 1916. Este UAV era controlado por radio desde tierra y sirvió como blanco aéreo de entrenamiento y como defensa contra los Zepelines.

En los últimos años, el uso de drones se ha extendido a otro tipo de aplicaciones civiles. Actualmente son utilizados para multitud de usos como pueden ser tareas de control de eventos, búsqueda de personas, vigilancia fronteriza, control de incendios forestales, revisión de infraestructuras, etc.

1.2. Definición del trabajo y objetivo

En la actualidad, el uso de los drones cuadricópteros en el entorno académico e industrial está en auge. Con la evolución de estos robots se ha conseguido dispositivos cada vez más pequeños y por lo tanto, más ligeros. Los drones de pequeño tamaño son conocidos como minidrones, y su reducido tamaño los hace aptos para trabajar en entornos interiores como pueden ser almacenes, pabellones deportivos, centros académicos, etc. En interiores, al no disponer de señal GPS, es necesario encontrar alternativas que permitan posicionar el dron para que pueda funcionar de forma autónoma.

En el presente trabajo se ha implementado un sistema de posicionamiento en interiores y control remoto para el minidrón Parrot-Mambo -PM- (Figura 1.2—1). Para el diseño, simulación e implementación de los controladores necesarios se ha utilizado las herramientas de Matlab/Simulink 2019.

Figura 1.2—1 - Minidrón Parrot Mambo

El objetivo general de este TFM es dotar al dron PM de un sistema de posicionamiento absoluto que permita el control remoto del mismo utilizando las herramientas de diseño e implementación de controladores de Matlab/Simulink.

Durante el desarrollo del TFM se ha trabajado en el diseño e implementación de un control de posición local y un sistema remoto, detallando en cada caso los resultados obtenidos, así como la problemática encontrada en cada una de las fases del trabajo.

1.3. Descripción del documento

En el capítulo segundo se describen los conceptos físicos de un cuadricóptero, el equipamiento hardware del minidrón Parrot Mambo utilizado y los recursos software disponibles en Matlab/Simulink.

En el capítulo tercero se muestra la contribución realizada al control y guiado del Parrot Mambo. Además, se muestras los resultados experimentales obtenidos a partir tanto de simulaciones como de ejecuciones con el minidrón.

En el capítulo cuarto se indican las conclusiones obtenidas del trabajo realizado con el minidrón Parrot Mambo. Las conclusiones están basadas en los resultados experimentales y su comparación con resultados en simulación.

En el capítulo quinto se enumeran posibles líneas de trabajo futuro planteadas a partir de las conclusiones obtenidas, así como en las posibilidades de mejora y limitaciones que se han encontrado en las herramientas utilizadas.

2. PARROT MAMBO: FUNDAMENTOS TEÓRICOS Y HERRAMIENTAS DE CONTROL

Es este capítulo se describe, en primer lugar, el funcionamiento físico de un dron, cuáles son las tareas a realizar por el control del dron y a continuación, los componentes hardware del dron Parrot Mambo.

2.1. Conceptos de modelado y control de cuadricópteros

Un cuadricóptero es un dron de tipo multirrotor con cuatro rotores. Para que el cuadricóptero se eleve los rotores deben ejercer una aceleración que supere a la gravedad. Esta aceleración que ejercen los motores es conocida como ***Throttle*** y siempre es perpendicular al plano del dron, es decir, al plano que contiene los cuatro motores en el caso del cuadricóptero. Controlando esta aceleración es posible controlar la altura del dron como se representa en la Figura 2.1—1.

Figura 2.1—1 – Aceleración vertical: Throttle

Como se ha indicado anteriormente, la aceleración *Throttle* siempre es perpendicular al plano del dron, de este modo si el plano del dron es completamente vertical el dron se elevará o descenderá en función del empuje que realicen los motores. En cambio, inclinando el plano del dron, la aceleración ejercida por los motores (*Throttle*) puede descomponerse en una componente vertical que mantiene la altura del dron y en otra horizontal que genera un desplazamiento lateral como se indica en la Figura 2.1—2.

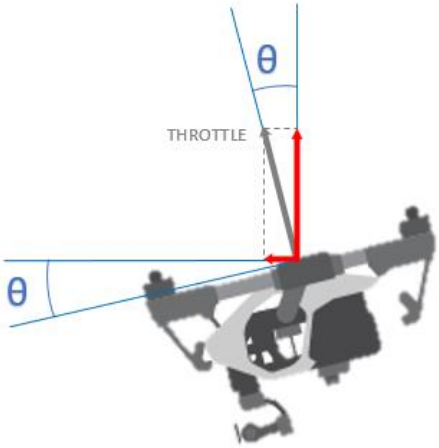


Figura 2.1—2 - Ángulo de desplazamiento

El ángulo de desplazamiento mencionado se denomina ***Pitch*** si se produce en el eje longitudinal del dron o ***Roll*** si se produce en el eje transversal del dron. Los ángulos Pitch y Roll producen desplazamientos frontales o laterales respectivamente. Véase Figura 2.1—3.

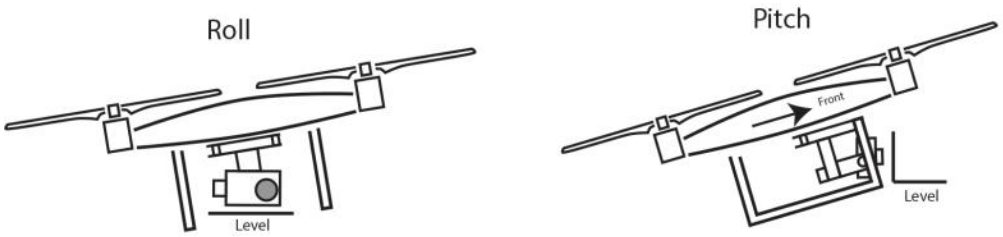


Figura 2.1—3 - Ángulos Roll y Pitch

Por último, la orientación del dron se evalúa con el ángulo ***Yaw***, definido como rotación intrínseca alrededor del eje vertical perpendicular al plano que forman los rotores del dron. Véase Figura 2.1—4.

Figura 2.1—4 Ángulo Yaw

El sentido de giro de los cuatro rotores es como se muestra en la Figura 2.1—5. De esta forma, se compensa el par de torsión que produce el giro de las hélices.

Figura 2.1—5 - Sentido de giro de los rotores

Para cambiar la orientación del dron, es necesario aumentar la velocidad de giro del par de rotores que giran en ese sentido y que están situados en una diagonal como se indica en la Figura 2.1—6.

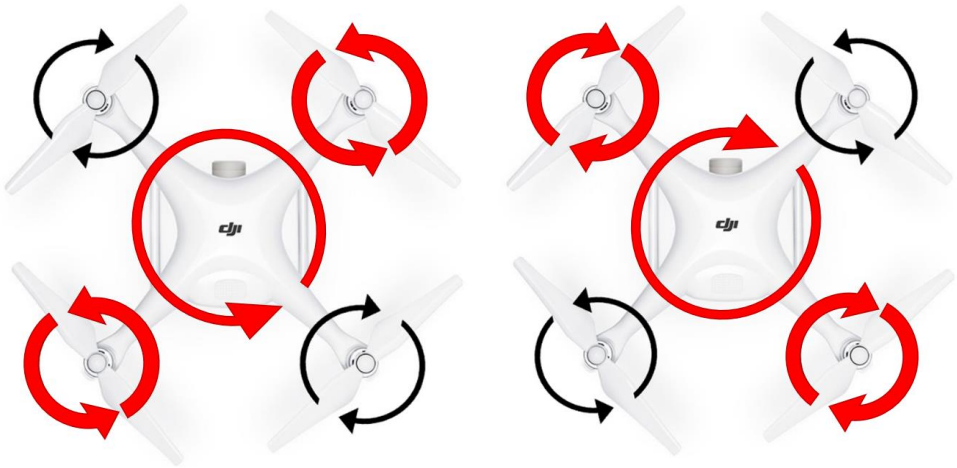


Figura 2.1—6 - Sentido de giro Yaw

2.1.1.Control del dron

En el modelo inicial proporcionado por Parrot, el control se realiza mediante cuatro controladores desacoplados: un control para cada uno de los ángulos de navegación (Pitch, Roll, Yaw) y otro para el control de altitud (Throttle).

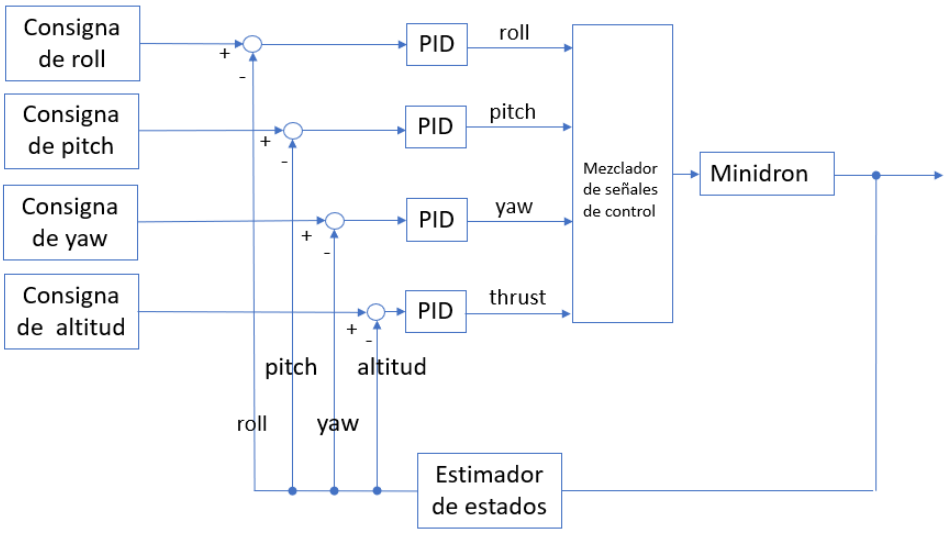


Figura 2.1—7 - Control simplificado de un dron

Según se muestra en la Figura 2.1—7, las referencias del control mostrado son los ángulos de navegación (Pitch, Roll, Yaw) y la altura. Para realizar el control en posición (coordenadas X,Y), se añaden dos controladores (Controlador de posición) como se muestra en la Figura 2.1—8, que serán los encargados de generar la consigna de pitch y roll en función del error de posición.

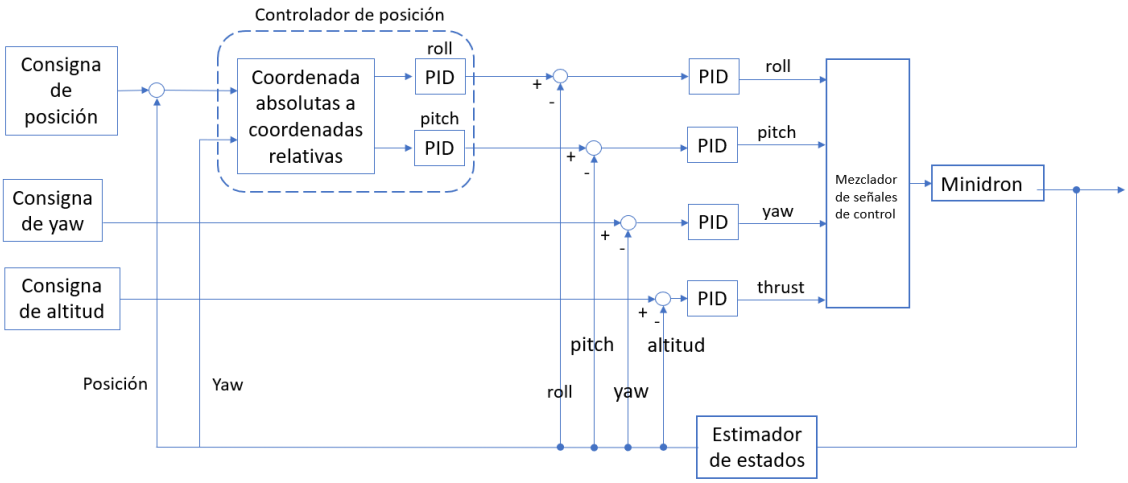


Figura 2.1—8 - Control en posición de un dron

2.2. Parrot Mambo. Modelado y control básico

En este apartado se describe el hardware disponible en el minidrón Parrot Mambo. En primer lugar, se describen los sensores integrados en el dron:

- Unidad de medida inercial -IMU- (acelerómetro de 3 ejes y giroscopio de 3 ejes). Permite conocer las aceleraciones lineales y angulares, a partir de las cuales es posible calcular velocidades, posición y orientación del dron.
- Sensor de ultrasonido -sónar- con alcance de 4 m. Permite conocer la altitud relativa con respecto a una superficie u obstáculo situado en la parte inferior del minidrón cuando se encuentra en vuelo.
- Sensor de presión -barómetro-. Permite conocer incrementos de altitud del dron.
- Cámara vertical de 60 fps situada en la parte inferior del dron. Esta cámara se utiliza para calcular la velocidad del dron analizando el flujo óptico. El flujo óptico es el patrón del movimiento aparente de los objetos, superficies y bordes en una escena causado por el movimiento relativo entre un observador (un ojo o una cámara) y la escena. La imagen captada por la cámara es accesible mediante la librería de Simulink suministrada por Parrot (véase apartado 3.2). El sensor de la cámara tiene una resolución de 120 x 160 píxeles.

En la Figura 2.2—1 se muestra la vista inferior del dron Parrot Mambo, en al que se puede ver dónde están situados dos de los sensores: el sensor de ultrasonido o sónar y la cámara vertical.

Figura 2.2—1 – Vista inferior del dron Parrot Mambo

Además de los sensores indicados, el dron dispone de los siguientes componentes:

- Batería LiPo 550mAh que aporta una autonomía de vuelo de aproximadamente 9 minutos.
- Conexión Bluetooth Low Energy -BLE- versión 4.0 que permite establecer una red de área personal (Personal Area Network -PAN-) inalámbrica entre el PC y el dron. La conexión BLE tiene tres funciones: cargar el código generado mediante el software de Matlab/Simulink, monitorizar y controlar las señales internas del dron. La conexión BLE 4.0 tiene una velocidad de transmisión de datos de 32 Mb/s.

2.2.1.Modelo Simulink del Parrot Mambo

En este apartado se incluye una descripción del software para minidrones de Parrot utilizado en el TFM. El software es suministrado por Parrot a través de los repositorios de Matlab.

2.2.1.1. Simulink Support Package for Parrot Minidrones

Para trabajar con el dron Parrot Mambo se ha utilizado el Add-on de Matlab Simulink Support Package for Parrot Minidrones. Este Add-on es suministrado por Parrot a través del repositorio Matlab 2017a y versiones posteriores, como se muestra en la Figura 2.2—2.

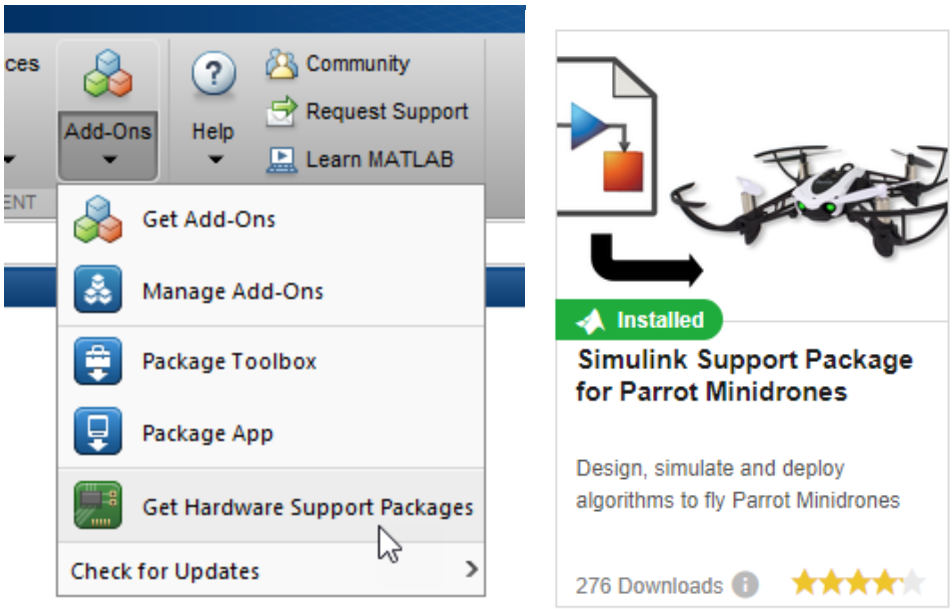


Figura 2.2—2 – Simulink Support Package for Parrot Minidrones

Simulink Support Package for Parrot Minidrones permite diseñar e implementar los algoritmos de control de vuelo para minidrones Parrot. Los algoritmos de control pueden acceder a la

información generada por los sensores del dron (véase apartado 2.2) y generar la señal de velocidad de giro de los rotores.

A través de la conexión Bluetooth Low Energy -BLE- se descarga el código diseñado en Matlab en el dron, donde se ejecuta de forma local el algoritmo de control. La conexión BLE permite también comunicar el dron con el PC remoto durante la ejecución y así monitorizar en tiempo real las señales internas del control implementado. También es posible comunicar el algoritmo que se ejecuta en el dron con otros algoritmos ajenos al dron y que se están ejecutando en el PC remoto. De esta forma es posible, por ejemplo, enviar comandos al dron como pueden ser unas coordenadas destino.

La conexión BLE 4.0 utilizada tiene una velocidad de trasmisión de datos de 32 Mb/s. Esta velocidad limita la cantidad de información que quiere monitorizarse desde el PC remoto y puede no ser suficiente si se quiere, por ejemplo, transmitir la imagen obtenida por la cámara del dron en tiempo real a un PC remoto para su visualización.

2.2.1.2. Descripción del modelo

El Add-on Simulink Support Package for Parrot Minidrones cuenta con un modelo (Figura 2.2—3) para poder simular o ejecutar vuelos con el dron.

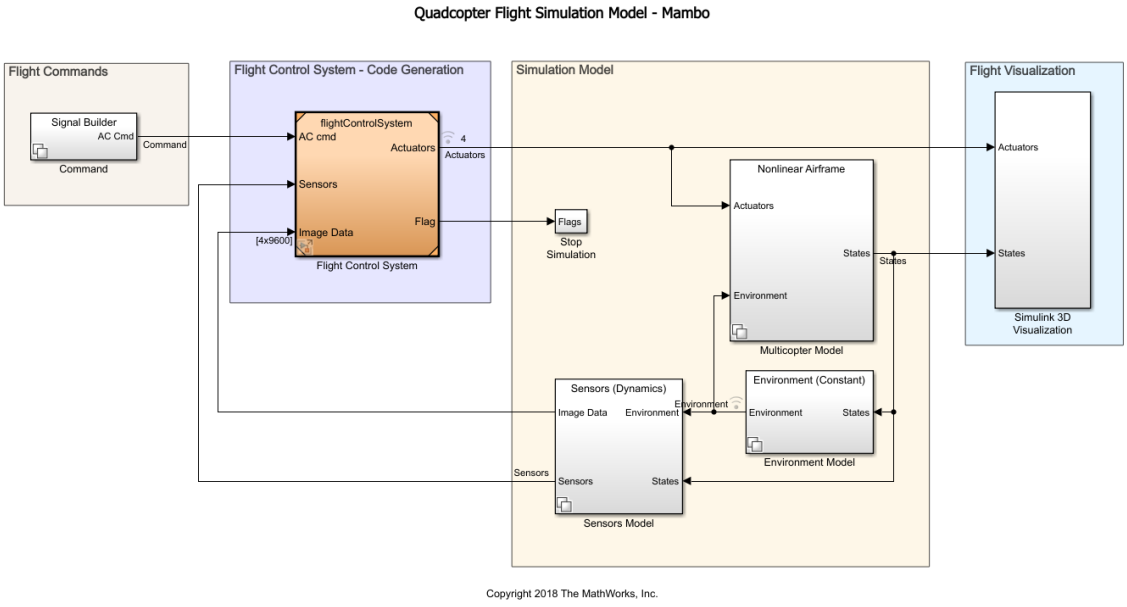


Figura 2.2—3 - Algoritmo inicial de control de vuelo para Parrot Mambo

Tal y como puede observarse en la figura, el modelo está dividido en cuatro subsistemas:

- Flight Commands: En este subsistema se genera la referencia o trayectoria de forma externa al dron. Este bloque no se utiliza en el presente trabajo, puesto que no es necesario en un funcionamiento autónomo del dron.
- Flight Control System: Este subsistema contiene toda la lógica de control del dron. A partir de las referencias y de la información sensorial calcula las señales de actuación del dron.
- Simulation model: Este subsistema se encarga de simular el comportamiento del dron, el entorno de trabajo y el sensado del conjunto dron-entorno.
- Flight Visualization: Este subsistema es el encargado de generar de forma visual el comportamiento del dron, es decir, genera una visualización gráfica del dron y su entorno durante la simulación, tal y como muestra la Figura 2.2—4.

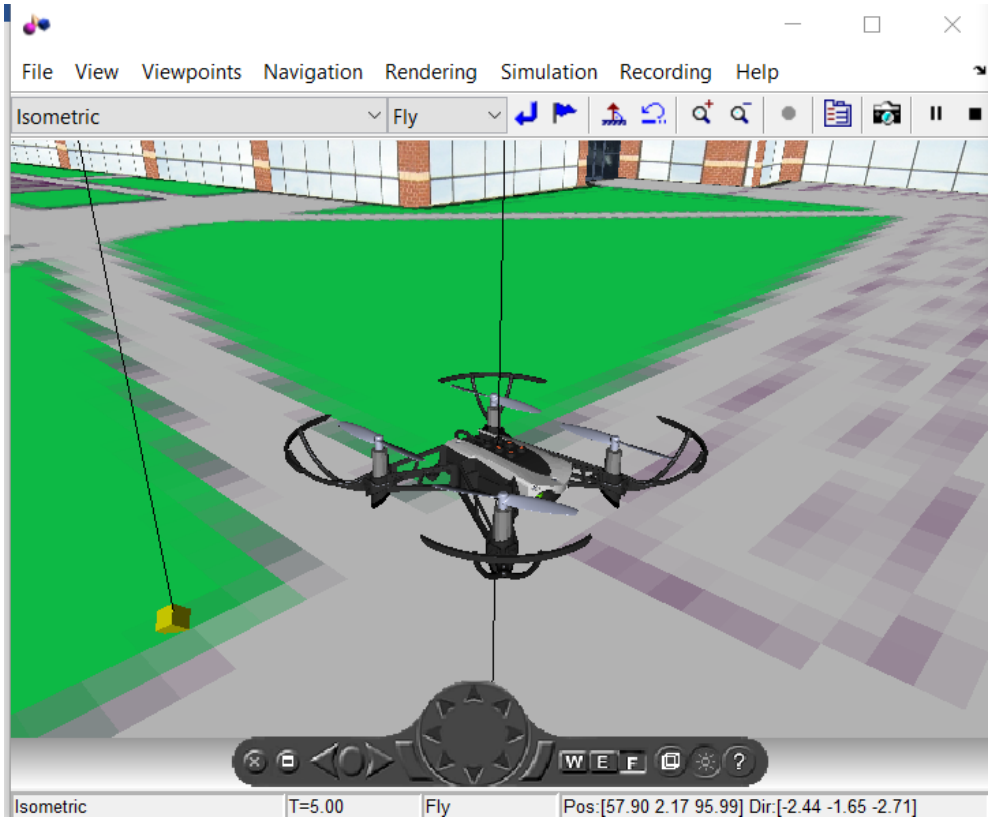


Figura 2.2—4 - Visualización de simulación de vuelo Parrot Mambo

Flight Control System (FCS)

A continuación, se profundiza en el sistema de control de vuelo (Flight Control System –FCS-). Este sistema opera, tanto en simulación como en ejecución real, una vez conectado de forma inalámbrica el dron con el PC de diseño y supervisión remota.

El FCS contiene los dos bloques mostrados en la Figura 2.2—5, que funcionan a distinta frecuencia (periodo de muestreo):

- Image Processing System: Este bloque funciona con un periodo de 200ms, es el encargado de procesar la información obtenida de la cámara inferior del dron.
- Control System: Este bloque funciona con un periodo de 5ms con el que se actualizan los algoritmos de control y de estimación de estados.

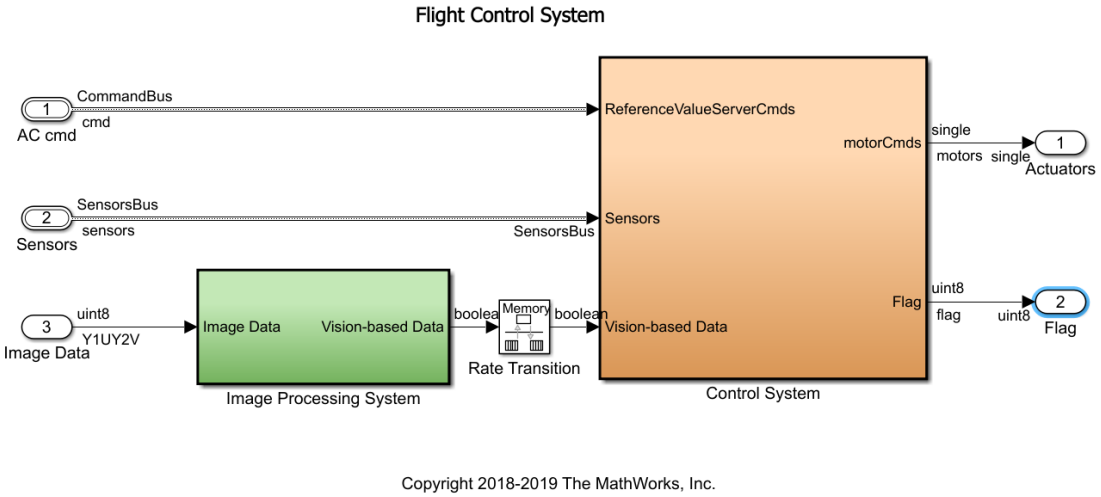


Figura 2.2—5 - Contenido del FCS

El bloque Rate Transmision es el encargado de intercambiar información entre dos subsistemas que operan a distinta frecuencia de muestreo, manteniendo la última información obtenida del procesado de imagen hasta disponer de nueva información de la cámara.

3. CONTRIBUCIÓN AL CONTROL Y GUIADO DEL PARROT MAMBO

En este apartado se describe la aportación realizada en el TFM al control del minidrón Parrot Mambo. Se diferencian tres aportaciones:

- Modificaciones realizadas en el controlador de posición para obtener el comportamiento esperado.
- Descripción de una solución de procesado de la imagen captada por la cámara vertical del dron basada en la detección y decodificación de patrones en el suelo. Se muestran los resultados obtenidos, así como problemas encontrados y la propuesta de mejora.
- Modificaciones en el estimador de posición para corregir el error acumulado mediante la información recibida de la cámara.

Por último, se muestran los resultados obtenidos en pruebas tanto en simulación como en ejecución con el dron.

3.1. Ajuste de controladores

En el siguiente apartado se describe las modificaciones realizadas en las ganancias del controlador de posición proporcionado por Parrot para conseguir el comportamiento idóneo para la aplicación buscada.

3.1.1. Controlador de posición

El controlador del que se parte en el modelo utilizado en este proyecto realiza un control en posición absoluta, es decir, la señal de error es la diferencia entre la referencia y la posición estimada en coordenadas X e Y.

$$P_{err} = \begin{bmatrix} X_{err} \\ Y_{err} \end{bmatrix} = \begin{bmatrix} X_{ref} \\ Y_{ref} \end{bmatrix} - \begin{bmatrix} X_{est} \\ Y_{est} \end{bmatrix}$$

A la salida del controlador se obtiene las señales que alimentarán los 4 rotores. En la Figura 3.1—1 se muestra un esquema simplificado del control de posición.

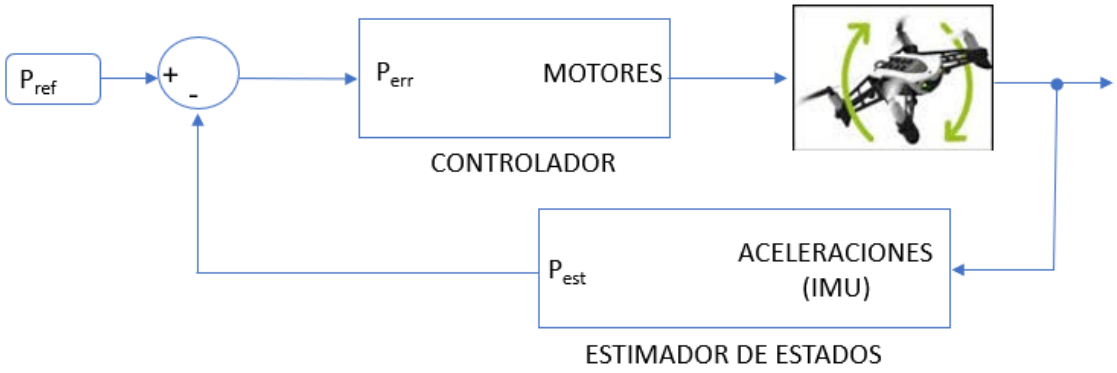


Figura 3.1—1 - Esquema control en posición absoluta

Como se muestra en la Figura 3.1—2, el controlador puede dividirse en 3 etapas:

- La primera etapa transforma el error de posición en una referencia de *pitch* y *roll*.
- La segunda etapa calcula el error de *pitch (θ)* y *roll(ϕ)* y genera las señales de actuación τ_pitch y τ_roll.

$$O_{err} = \begin{bmatrix} \theta_{err} \\ \phi_{err} \end{bmatrix} = \begin{bmatrix} \theta_{ref} \\ \phi_{ref} \end{bmatrix} - \begin{bmatrix} \theta_{est} \\ \phi_{est} \end{bmatrix}$$

- La tercera etapa será la encargada de combinar las señales de actuación τ_pitch y τ_roll con las señales homólogas obtenidas de los controladores de altitud (τ_trottle) y orientación (τ_yaw). Estos últimos representados en gris.

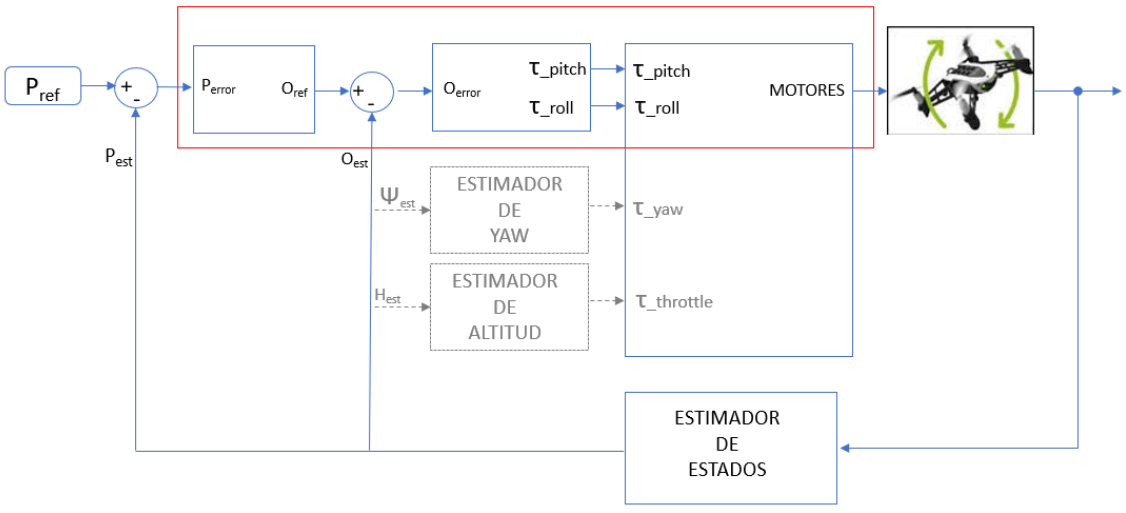


Figura 3.1—2 - Contenido del controlador

3.1.2.Ajuste del controlador

Para poder obtener el mayor número de medidas válidas de la cámara del dron, es necesario que el movimiento del dron sea lo más suave posible, reduciendo en todo lo posible las oscilaciones innecesarias. Para conseguir esto, se ha buscado modificar el compartimiento del dron para conseguir un movimiento con mayor amortiguamiento. En la Figura 3.1—3 se muestra la señal de error inicial (rojo) y la señal de error modificada (azul) con mayor amortiguamiento.

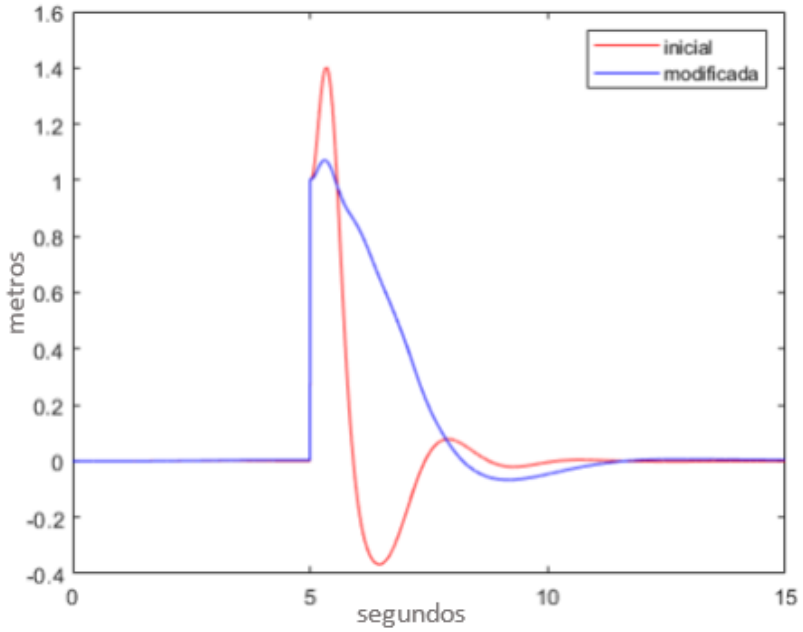


Figura 3.1—3 – Señal de error de posición ante entrada escalón

Para realizar el ajuste del controlador de manera segura, se utiliza el modelo de simulación disponible en el Simulink Support Package proporcionado por Parrot.

3.1.2.1. Estudio del controlador y su respuesta

En primer lugar, se describe la primera etapa del controlador de posición. Este controlador consiste en la combinación de dos controladores proporcionales en paralelo como se muestra en la Figura 3.1—4.

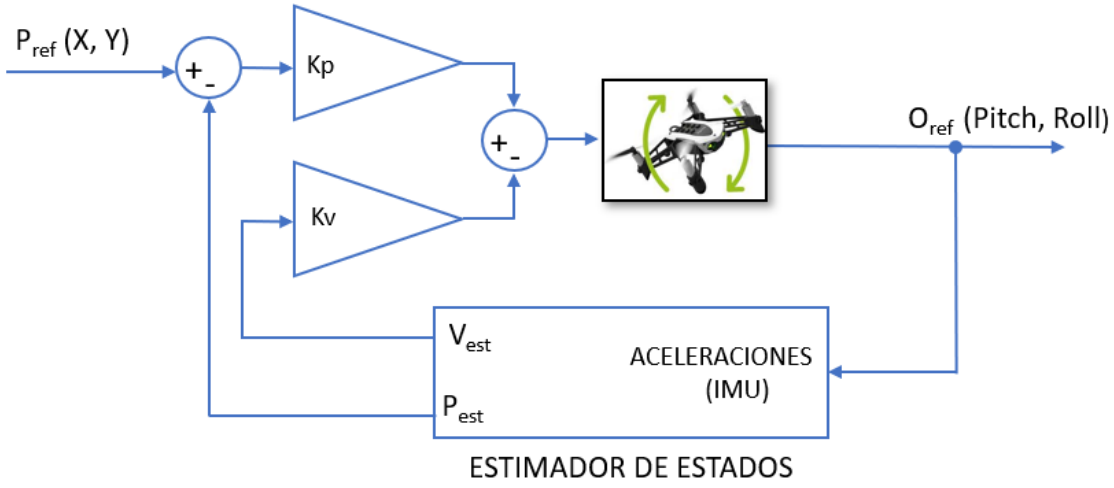


Figura 3.1—4 – Primera etapa del controlador: Conversión referencia en posición a referencia en pitch/roll

Uno de los controladores utiliza la señal de error de posición (metros) entre la consigna de posición y la posición estimada.

$$\mathrm{P_{err}} = P_{ref} - P_{est}$$

El otro controlador tiene como entrada el error de velocidad entre la consigna, y la velocidad estimada del drone, en este trabajo al realizar una estrategia de path-following la velocidad de referencia es nula:

$$v_{err} = v_{ref} - v_{est} = -v_{est}$$

Para el estudio, se ha trabajado primero en simulación utilizando el modelo proporcionado en el software de Parrot. Para el estudio de la respuesta del controlador, se introduce una señal escalón como señal de entrada del eje x, es decir, pasar de la posición inicial (0,0) a las coordenadas (1, 0).

A continuación, se muestra la señal de error de posición obtenida en las simulaciones para comprender el comportamiento del modelo dron con el controlador proporcionado por Parrot por defecto.

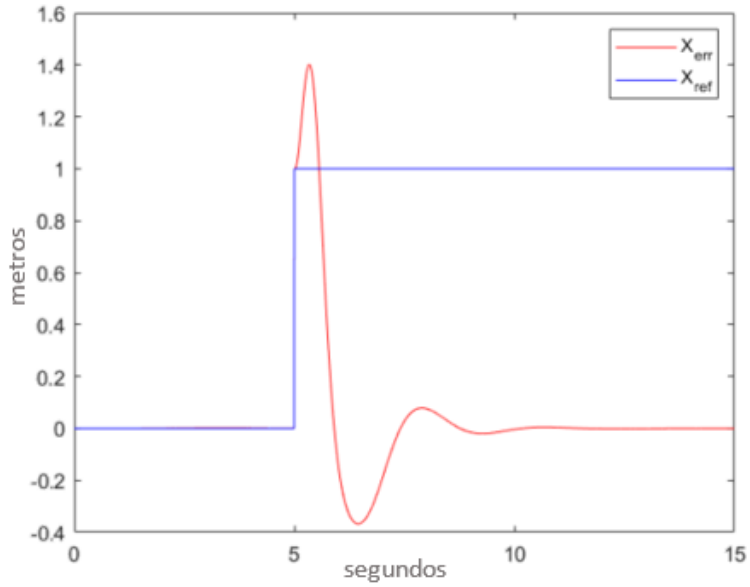


Figura 3.1—5 – Error de posición (rojo) ante estrada escalón (azul). Solución inicial

En la Figura 3.1—5, se observa que el dron responde de manera muy rápida, lo que provoca que se produzcan oscilaciones cuando el error llega a cero. Durante este tiempo, la información de la cámara no es válida puesto que el dron no se encuentra totalmente horizontal. Esto motiva la modificación del controlador de posición para conseguir un comportamiento más amortiguado (ver Figura 3.1—6). De esta forma, el dron llega a las coordenadas objetivo con menor inclinación sobre el plano horizontal, obteniendo antes una imagen válida y estable de la cámara.

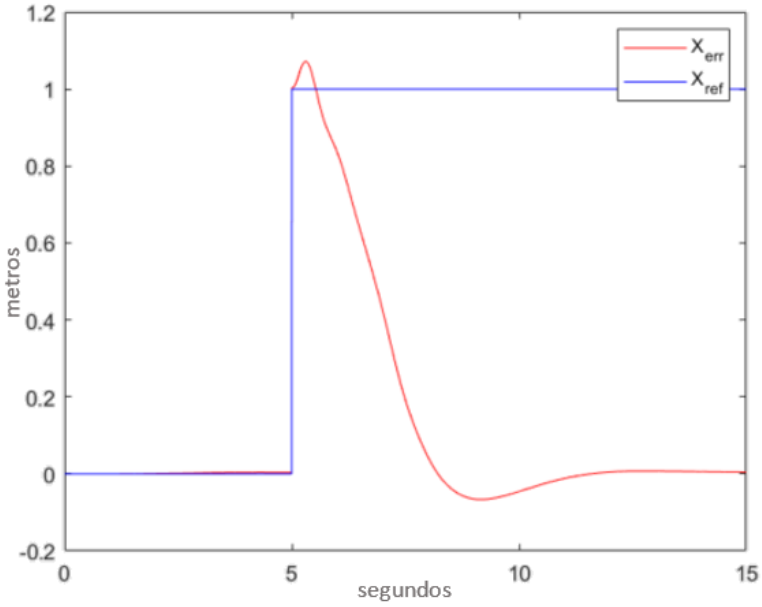


Figura 3.1—6 – Error de posición (rojo) ante estrada escalón (azul). Solución modificada

Se observa que cuando se produce el salto en el error de posición debido al cambio de referencia de trayectoria, el error de posición crece en un primer momento para después volver a bajar. Se ha comprobado que este comportamiento es debido a un error en el entorno de simulación proporcionado. Para comprobar el comportamiento del dron en ejecución se realiza la misma prueba con el dron.

En la Figura 3.1—7 se muestra una comparativa del error de posición en ejecución y en simulación. En este caso, al recibir la entrada escalón en la referencia, se observa que el error decrece inmediatamente después. Además, se observa que el comportamiento en ejecución no coincide exactamente con la simulación, por lo que los resultados en simulación deben ser comprobados en ejecución.

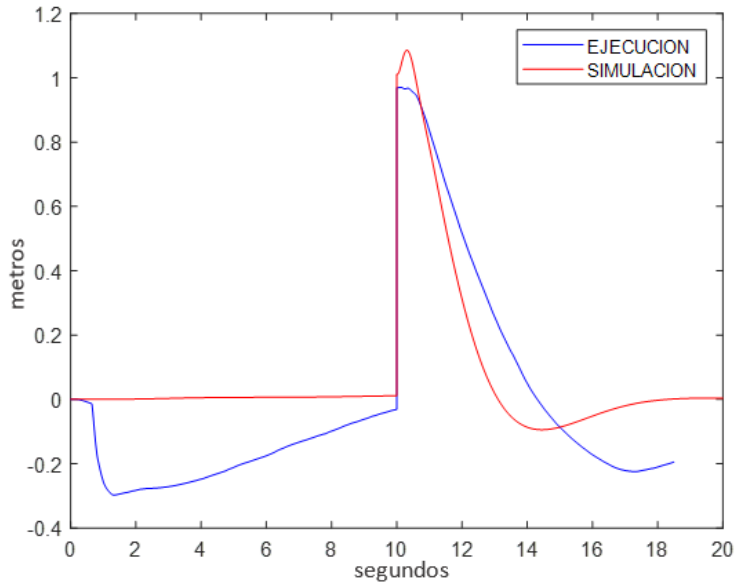


Figura 3.1—7 – Comparación de señales de error de posición en simulación (rojo) y ejecución (azul).

Tras las pruebas realizadas, el valor de las ganancias modificadas se muestra en la siguiente tabla, al tratarse el modelo del dron de un modelo muy complejo, este ajuste se ha realizado desacoplando la relación entre el pitch y el roll para de esta forma simplificar el problema:

	Kp_x	**Kp_y**	**Kv_x**	**Kv_y**
Valor inicial	-0.24	0.24	0.1	-0.1
Valor utilizado	-0.1	0.1	0.15	-0.15

3.2. Procesado de imágenes

El objetivo principal de este TFM es diseñar e implementar en el minidrón Parrot Mambo una solución que permita seguir una trayectoria de manera autónoma, siendo capaz la solución diseñada de compensar los errores acumulativos de posicionamiento del IMU. Para ello, se utiliza el reconocimiento con visión artificial de marcas situadas en el suelo a modo de balizas pasivas cuya posición es conocida con exactitud.

Para corregir la posición estimada del dron de forma autónoma, se ha utilizado la imagen obtenida por la cámara vertical integrada en el dron. Las imágenes capturadas por la cámara están disponibles en el subsistema de procesamiento de imagen a través de la entrada *Image Data* mostrada en la Figura 3.2—1.

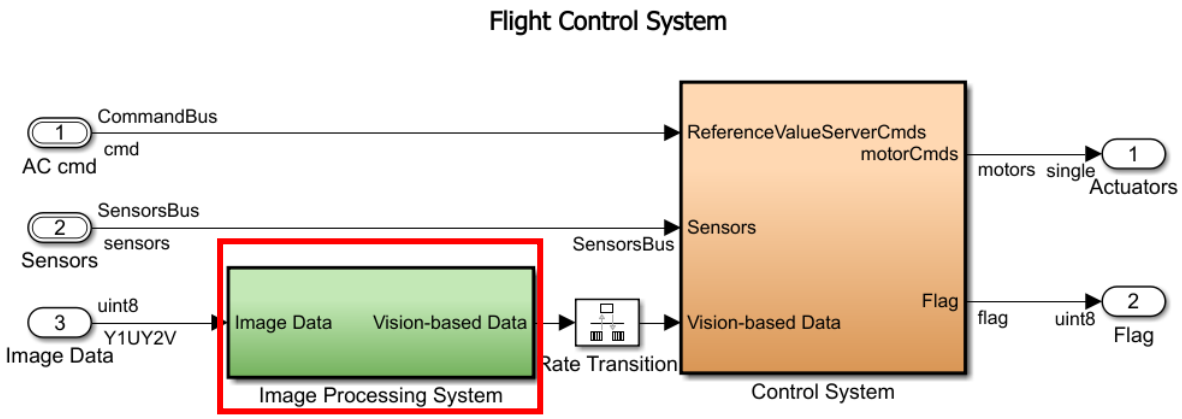


Figura 3.2—1 - Subsistema de procesamiento de imagen (resaltado en rojo)

La imagen se obtiene codificada en formato Y1UY2V. Para trabajar en formato RGB con la imagen de la cámara se utiliza el conversor de imagen de la librería para minidrones Parrot de Simulink (ver Figura 3.2—2). A la salida de este bloque se obtiene la imagen en componentes RGB: rojo, verde y azul.

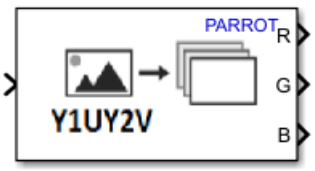


Figura 3.2—2 - Conversor de imagen PARROT incluido en la librería de Parrot

A partir de la imagen RGB, es posible analizar distintos patrones situados en el suelo que puedan utilizarse a modo de balizas fijas que aportan información al dron para corregir su posición.

A continuación, se describen dos soluciones utilizadas a modo de baliza fija, que sirven como referencia de posición conocida para corregir la posición estimada del dron. Posición que puede verse alterada por la acumulación de error que se produce con técnicas de posicionamiento basadas en el uso de la IMU.

3.2.1.Códigos de barras bidimensionales: marcas 2D

El primer método utilizado es el uso de códigos de barras bidimensionales, que es una forma de codificar información de manera gráfica en dos dimensiones en vez de barras paralelas, por lo que permiten incluir más información por unidad de área que los códigos de barras lineales. Existen numerosos tipos de códigos bidimensionales como se muestra en la Figura 3.2—3.

Figura 3.2—3 – Ejemplos de códigos de barras bidimensionales.

La elección del código bidimensional a utilizar depende de la información que se necesite codificar. Para este proyecto se ha decidido utilizar los propuestos por ArUco[1], mostrado en la Figura 3.2—4.

[1] http://www.uco.es/investiga/grupos/ava/node/26 (grupo de investigación "Aplicaciones de la Visión Artificial" (A.V.A) de la Universidad de Córdoba.)

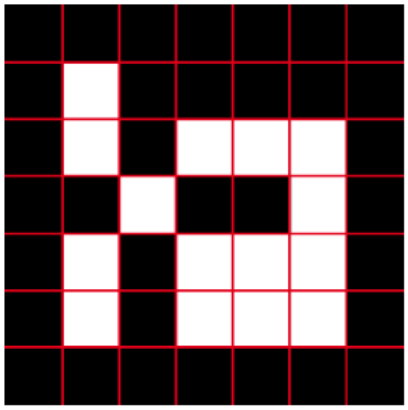


Figura 3.2—4 - Ejemplo marca 2D ArUco

Cada marca contiene un código que consiste en 5 palabras de 5 bits cada una. La codificación empleada es una modificación del código de Hamming. Consiste en que cada palabra contiene 2 bits de información y 3 bits de paridad para la detección de error como se muestra en la Figura 3.2—5. En total es posible generar 1024 marcas distintas.

Figura 3.2—5 - Bits de información y bits de paridad.

Cada combinación posible de los 2 bits de información tiene asignada una combinación de los 3 bits de paridad. En la Figura 3.2—6 se muestran las cuatro únicas combinaciones válidas que puede contener cada una de las filas. Las celdas negras se corresponden con un '0' lógico, mientras que las celdas blancas con un '1' lógico. Los bits de paridad se muestran resaltados en color rojo y los bits de información en verde.

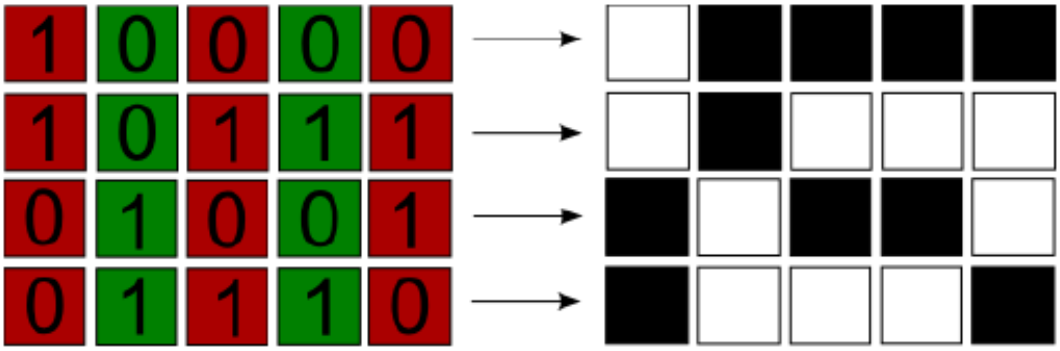


Figura 3.2—6 - Combinaciones posibles de cada palabra de información.

Como ejemplo de marca utilizada, en la Figura 3.2—7 se muestra la marca correspondiente al valor 101.

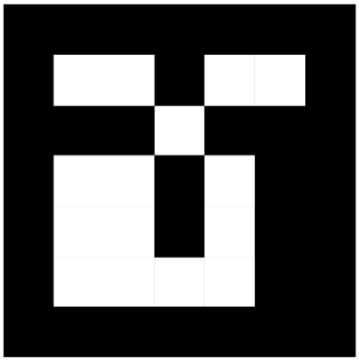


Figura 3.2—7 - Marca utilizada

3.2.1.1. Detección de la marca

Para la detección de la marca se ha utilizado la función de Matlab *regionprops*[2]:

```
stats = regionprops(BW,properties)
```

Esta función devuelve un array (`stats`) de N elementos tipo *struct*, siendo N el número de regiones o contornos que la función ha encontrado en la imagen binaria especificada en su

[2] Documentación de Mathworks: https://es.mathworks.com/help/images/ref/regionprops.html

primer argumento (`BW`). El número de campos de cada elemento tipo *struct* depende de las propiedades especificadas en su segundo argumento (`properties`).

Para este proyecto se han utilizado las propiedades:

- 'Area': Número real de píxeles en la región, devueltos como escalares. Se utilizará para aplicar un primer filtrado con objeto de descartar áreas demasiado pequeñas.
- 'Centroid': Coordenadas cartesianas en píxeles del centro de masa de la región. Indicará las coordenadas de la marca en caso de ser encontrada.
- 'Extrema': Puntos extremos en la región, devueltos como una matriz de 8 por 2 siendo cada punto coordenadas en píxeles de cada uno de los puntos indicados en la Figura 3.2—8. Se utiliza para delimitar el contorno de región encontrada, necesario para analizar si la región encontrada es la marca buscada y en caso positivo, decodificarla.

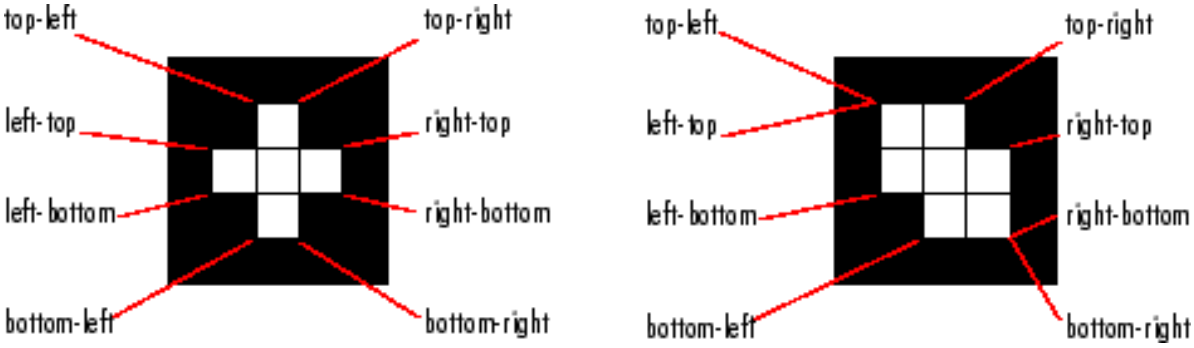


Figura 3.2—8 - Extremos de una región o contorno.

A continuación, se indica la línea de código utilizada:

```
reg=regionprops(BW_image,'Area','Centroid','Extrema');
```

Una vez identificadas las regiones de la imagen obtenida de la cámara del dron, se analiza cada una de ellas según los siguientes criterios:

- Área mayor que 100 píxeles cuadrados.
- Son cuadriláteros: Se aproximan los extremos obtenidos de regionprops a polígonos utilizando la función polyshape de Matlab. Esto permite desechar las regiones cuyo número de vértices sea distinto de cuatro.

Por último, se evalúan las regiones que cumplan las condiciones anteriores para identificar las marcas ArUco. Para ello se evalúan las siguientes condiciones:

- Los bordes son cuadrados de color negro.
- Se comprueban los bits de paridad de cada una de las filas

Si se cumplen las condiciones anteriores, la región evaluada es una marca válida y por tanto es posible decodificar el valor de la marca. En este caso, el valor de la marca utilizada es el 101.

Resultados obtenidos:

En la Figura 3.2—9 se muestra a la izquierda un fotograma captado por la cámara del dron, mientras que en la imagen de la derecha se observa la marca detectada por el código y resaltada en color azul, además se muestra la posición del centroide de la marca resaltada en rojo. Es importante tener en cuenta que las coordenadas del centroide de la marca son las coordenadas en pixeles, por lo que habrá que calcular su equivalencia a metros en función de la altura en la que se encuentra el dron. Esta altura es calculada por el estimador de estados del dron a partir de la información obtenida de los sensores del dron (IMU, barómetro y sónar).

Figura 3.2—9 - Detección de la marca (sombreado azul) y centroide (punto rojo)

Problemática de la solución

La función *regionprops* utilizada permite la generación de código, pero no es posible ejecutarla en el dron Parrot Mambo por limitaciones de procesamiento en el hardware local. Para solucionar esto, se ha optado por ejecutar el código de búsqueda de la marca en el PC conectado mediante bluetooth al dron. De este modo, el dron envía al PC remoto la imagen de la cámara mediante la conexión bluetooth, en el PC se procesa la imagen y envía de vuelta al dron la posición de las marcar encontrada en la imagen. Para esto, es necesario utilizar los bloques de comunicación de Simulink de la librería de Parrot.

3.2.1.2. Bloques de comunicación del "Simulink Support Package

La librería de Parrot para minidrones incluye los bloques de comunicación mostrados en la Figura 3.2—10, que sirven para enviar información entre el dron y el PC remoto. La comunicación se puede realizar utilizando los protocos TCP/IP o UDP. Al realizar un control en tiempo real sobre el lazo de comuniciones , se ha utilizado comunicación UDP debido a que de esta forma se evitan reenvios de información innecesarios en caso de que el canal de comunicaciones sufra pérdida de paquetes.

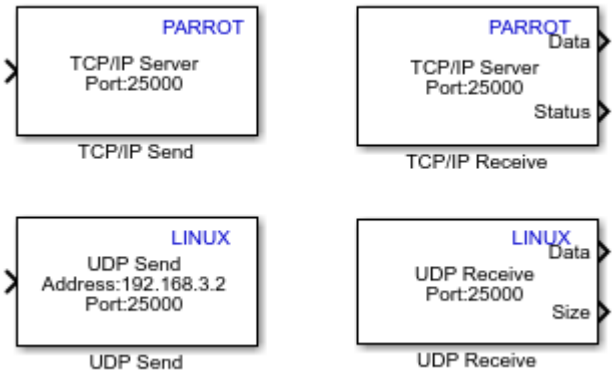


Figura 3.2—10 - Bloques de comunicación para minidrones Parrot

Como se muestra en el diagrama de la Figura 3.2—11, los bloques UDP Send y UDP Receive son los encargados de intercambiar la información entre el dron y el PC remoto donde se ejecuta el código encargado de encontrar las marcas en la imagen capturada por la cámara del dron.

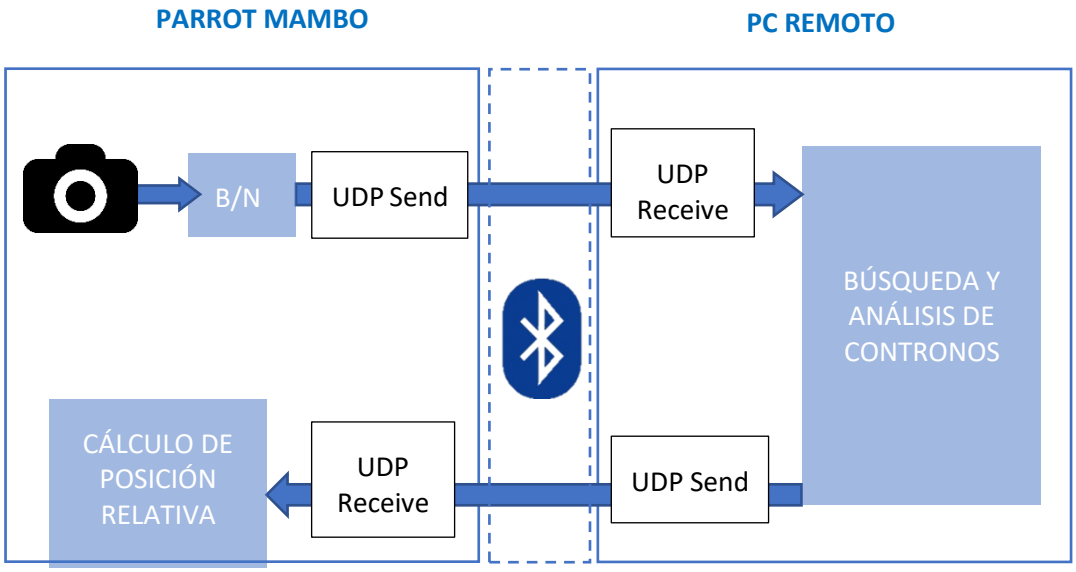


Figura 3.2—11 - Esquema de tratamiento de imagen en remoto

Para el correcto funcionamiento de los bloques de comunicación es necesario configurar el bloque emisor *UDP Send* con la dirección IP del equipo donde está ubicado el bloque receptor *UDP Receive*. La dirección IP del dron Parrot Mambo es la 129.168.3.1 mientras que la del PC remoto es la 192.168.3.2. También es necesario fijar el puerto utilizado en cada par emisor-receptor. En la Figura 3.2—12 se muestra la configuración del bloque emisor y en la Figura 3.2—13 se muestra la configuración del bloque receptor.

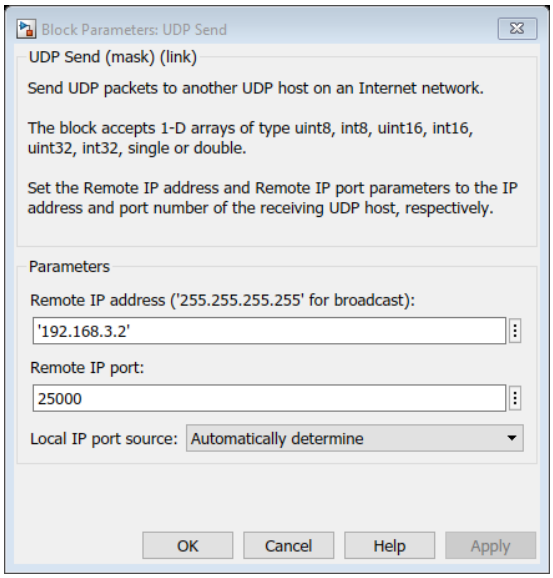


Figura 3.2—12 - Configuración de bloque UDP Send

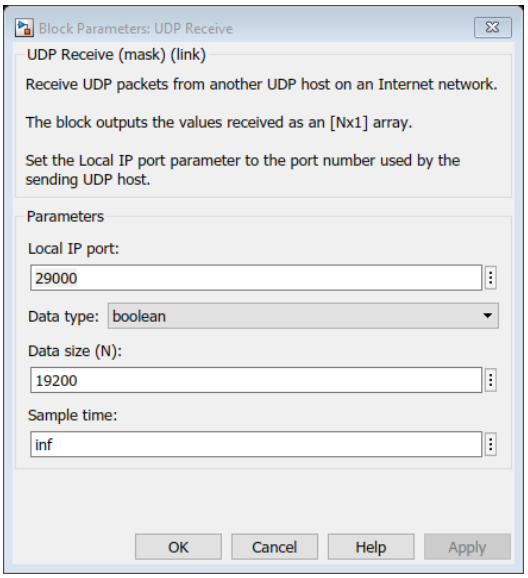


Figura 3.2—13 - Configuración de bloque UDP Receive

En el bloque receptor, es necesario fijar el tipo y el tamaño del vector que se espera recibir. En este caso, el vector que se envía es cada fotograma captado por la cámara previamente binarizado, por tanto, como la resolución de la cámara es 120 x 160 píxeles, se envía un vector de tipo 'boolean' con 19200 elementos.

3.2.1.3. Resultados y problemas encontrados

Durante los ensayos realizados con dron y PC remoto, se ha observado que existen retardos que afectan al funcionamiento del sistema completo. En la Figura 3.2—14, se muestra un ensayo en el que se envía al dron una señal sinusoidal generada en el PC remoto y se retorna la misma señal desde el dron al PC remoto para comparar las señales y medir el retardo producido.

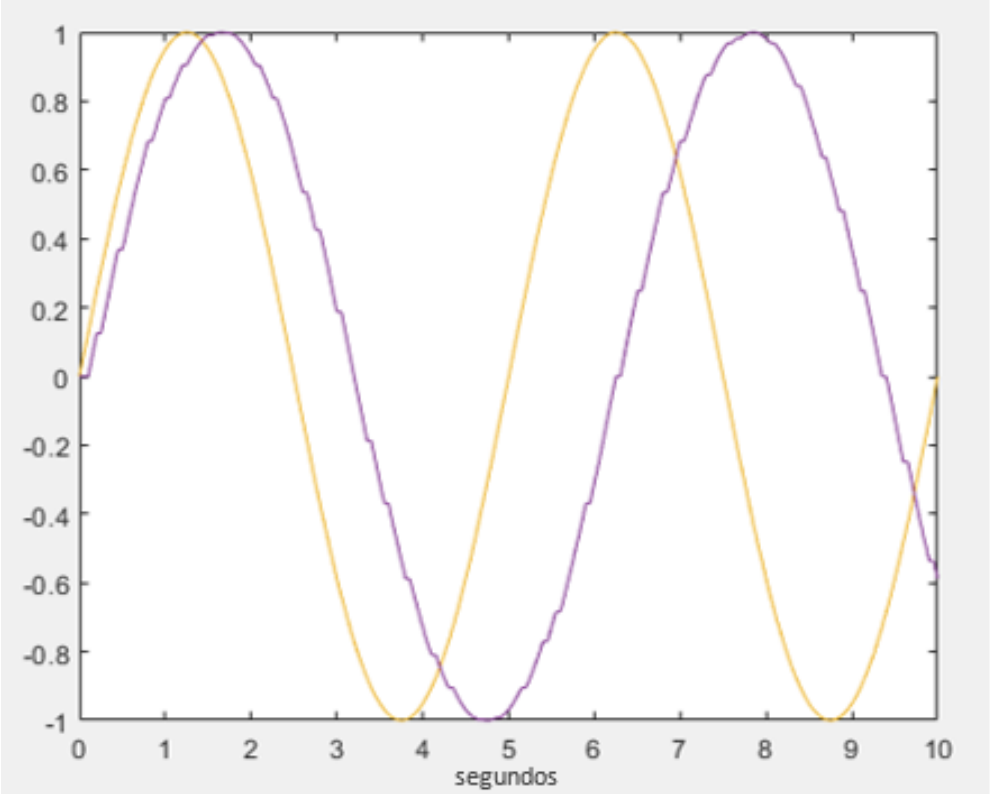


Figura 3.2—14 - Retardo en la comunicación vía bluetooth. Señal sinusoidal

En la Figura 3.2—15 se muestra el mismo ensayo con una señal rampa.

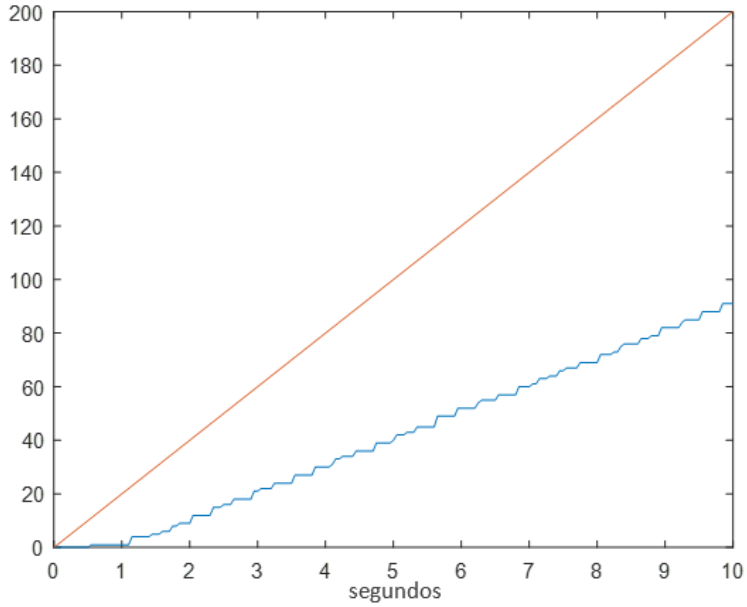


Figura 3.2—15 - Retardo en la comunicación vía bluetooth. Señal rampa

En ambas figuras se observa que el retardo no es constante, esto es debido a que el canal de comunicación bluetooth no es capaz de transmitir la información a la frecuencia de muestreo a la que funciona el sistema y se acumula a modo de *buffer*.

Es importante señalar que los bloques de comunicación *UDP Send* y *UDP Receive,* ejetutables en el dron, funcionan a la frecuencia de muestreo del sistema en el que se ejecutan. Al estar la frecuencia de muestreo del Parrot Mambo fijada por diseño, no es posible reducir el número de envíos de información realizados entre el dron y el PC para evitar que el retardo de las comunicaciones incremente según avanza la ejecución.

Para solucionar la problemática encontrada en el comportamiento del dron por los retardos introducidos durante la comunicación bluetooth, se estudia la opción de utilizar una solución de marca más simple que permita que el código de procesamiento de imagen pueda ejecutarse directamente en el dron. En el apartado 3.2.2 se describe la solución implementada.

3.2.2.Detección de contornos simples y corrección del error

Con objeto de evitar los problemas con los retardos introducidos por las comunicaciones con el PC remoto que ejecuta el código de procesamiento de imagen, se ha decidido buscar una solución de visión artificial que pueda ser ejecutada completamente de forma local en el dron.

Como se ha visto en el apartado 0, la función *regionprops* que se utiliza para identificar contornos en una imagen, no puede ser ejecutada en el minidrón Parrot Mambo, por lo que se opta por una solución basada en detección de marcas de un determinado color como las mostradas en la Figura 3.2—16.

Figura 3.2—16 - Marcas simples azules detectables por el algoritmo de visión artificial.

Como se muestra en la Figura 3.2—17, la solución consiste en dos etapas de procesamiento diferenciadas. Una primera etapa en la que se filtra la imagen por color y una segunda de detección de contornos en la imagen ya filtrada. A continuación, se detallan cada una de ellas.

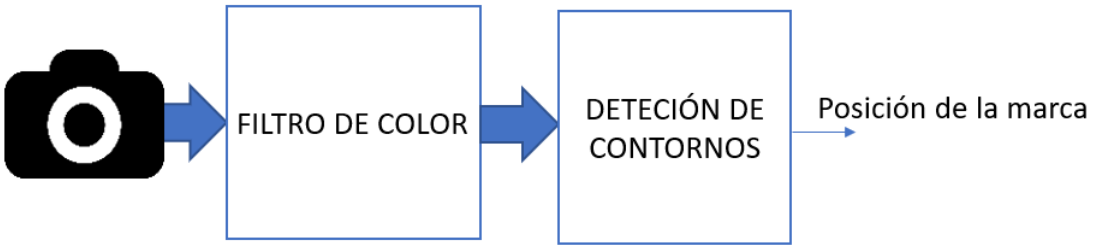


Figura 3.2—17 – Etapas de detección de contornos simples

Filtro de color: En la primera etapa se aplica una técnica de umbralización[3] para diferenciar el color buscado de la marca del resto de la imagen.

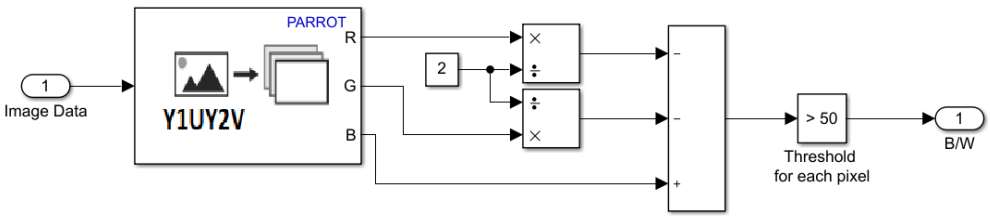


Figura 3.2—18 – Contenido de la etapa de filtrado de color

En el caso mostrado en la Figura 3.2—18 se filtra el color azul para detectar las marcas situadas en el suelo. En la Figura 3.2—19 se muestra el resultado del filtrado. En la imagen de la izquierda se muestra la imagen captada por la cámara y a la derecha la imagen filtrada para detectar la marca de color azul.

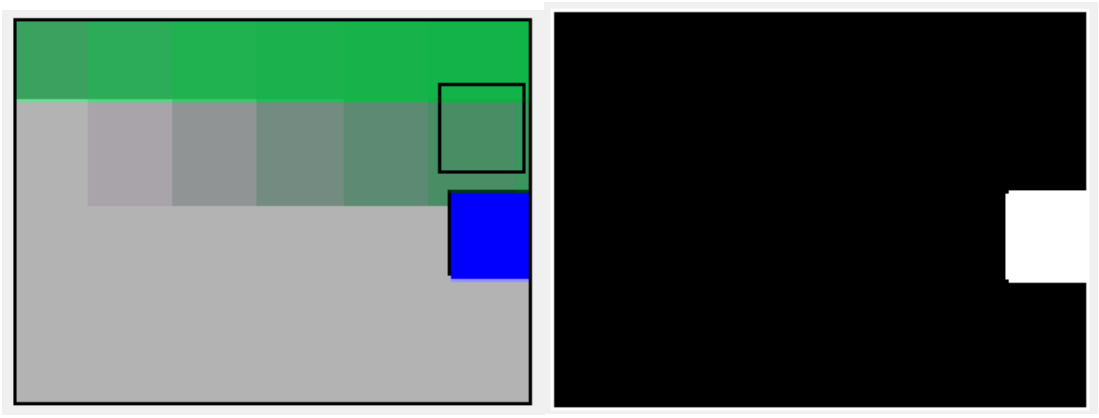


Figura 3.2—19 – Ejemplo de umbralización de la imagen. Resultados de simulación

Detección de contornos: esta etapa recibe la imagen filtrada o umbralizada y su función es la de detectar la marca y calcular su posición relativa con respecto al dron. En esta etapa se utiliza la librería *Computer Vision Toolbox*, que se muestra en la Figura 3.2—20. Esta librería proporciona

[3] La umbralización consiste en convertir una imagen a una nueva con sólo dos niveles o binaria, de manera que los objetos queden separados del fondo.

algoritmos y funciones para el diseño de aplicaciones de procesamiento de imagen y visión artificial.

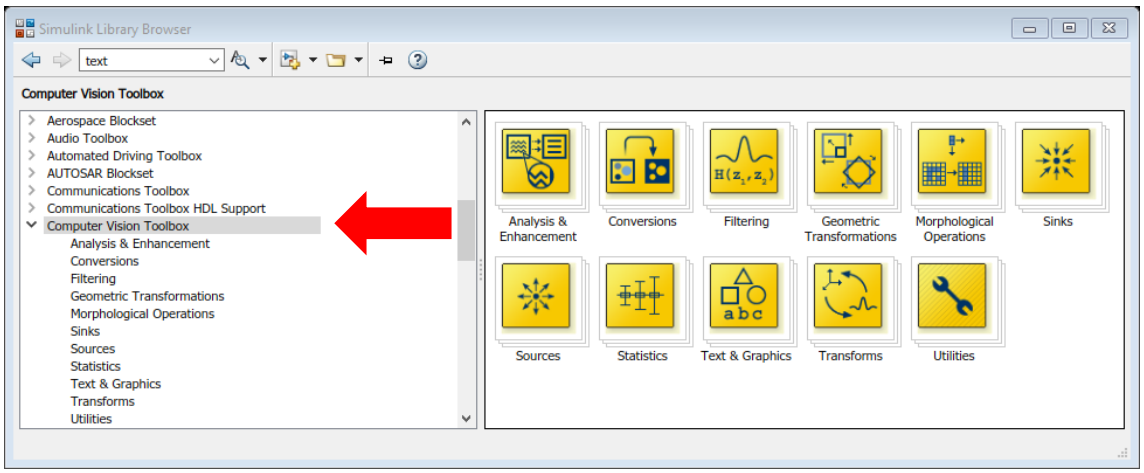


Figura 3.2—20 - Computer Vision Toolbox

El bloque utilizado en esta librería es *Blob Analysis* (ver Figura 3.2—21). Este bloque analiza la imagen umbralizada, es decir, en blanco y negro, y proporciona información referente a los contornos detectados en una imagen.

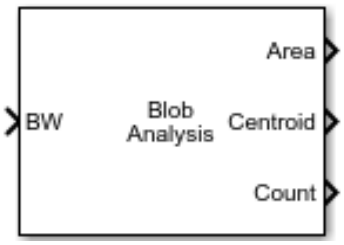


Figura 3.2—21 – Bloque Blob Analysis utilizado para detección de contornos.

Para este trabajo, se utiliza principalmente la siguiente información proporcionada por el bloque:

- Area: para descartar posibles contornos residuales demasiado pequeños para ser una de las marcas buscadas.
- Centroid: nos indicará las coordenadas del centroide de los contornos detectados.
- Count: Indica el número de contornos detectados. En este caso, para simplificar la solución y pueda ejecutarse sin problemas localmente en el dron, solamente se considera válida la marca cuando sólo se detecte una. Por lo tanto, este contador nos indica si la marca detectada es válida.

La solución de visión artificial proporciona al sistema de control del dron una señal binaria que se activa cuando se detecta la marca y otra señal que indica las coordenadas de la marca detectada.

3.3. Corrección de posición estimada

En este apartado se describe la solución implementada para corregir la posición estimada del dron cuando la cámara vertical capta una marca situada en el suelo.

3.3.1. Situación de partida

El estimador de estados proporcionado por Parrot está basado en cuatro estimadores que de manera simplificada pueden describirse de la siguiente manera:

- Estimador de orientación: calcula los ángulos pitch, roll y yaw a partir de la información de la IMU.
- Estimador de altitud: Calcula la altura a la que se encuentra el dron a partir de la información de IMU, sónar y barómetro.
- Estimador de velocidad: Calcula las componentes de velocidad del dron (dx, dy) a partir de la información de la IMU y el flujo óptico (optical flow) de la cámara vertical.
- Estimador de posición: Calcula la posición del dron (x, y) a partir de la información del estimador de velocidad.

En este trabajo se han implementado mejoras sobre la última etapa de estimación que corresponde con el estimador de posición. El estimador de posición proporcionado por Parrot, tal y como se muestra en la Figura 3.3—1, consiste en un integrador que proporciona la posición del dron a partir de la velocidad proporcionada por el estimador de velocidad.

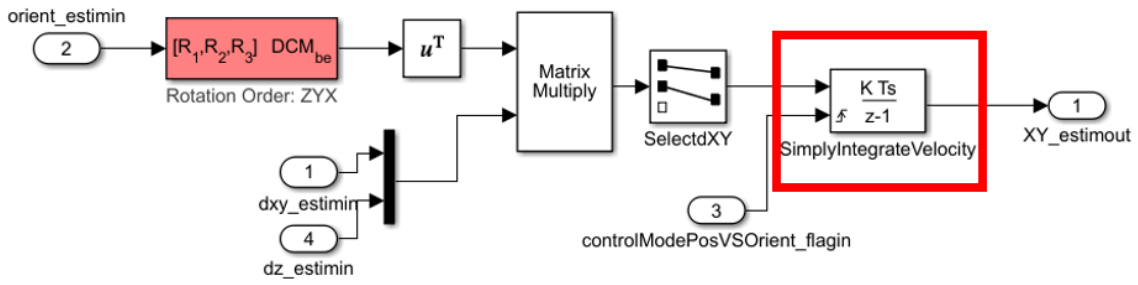


Figura 3.3—1 - Situación de partida del estimador de posición en modelo proporcionado por Parrot.

El objetivo es corregir posibles desviaciones en la estimación de la posición debidas al error que va acumulando la IMU y para ello se utiliza la información obtenida de la cámara. Como se ha explicado en el apartado 3.3, cuando la cámara vertical del dron detecta una de las marcas situadas en el suelo, es posible calcular la posición relativa del dron con respecto a la marca. Como se conoce la posición absoluta de cada una de las marcas colocadas en el suelo, es posible calcular la posición absoluta del dron. De esta forma, el dron recibirá de manera puntual y a través de la cámara información externa de su posición, que le servirá para corregir su posición estimada de manera interna (IMU).

3.3.2. Diseño del predictor-corrector

A partir de la información recibida del bloque de procesado de imagen (señal de detección de marca y coordenadas de la marca), se ha modificado el estimador de posición para que la posición estimada a partir de la información de aceleración que recibe de la IMU se corrija cada vez que se detecte una marca, cuya posición exacta es conocida. De esta forma, se realiza un posicionamiento absoluto del dron, actuando la marca como baliza pasiva.

Para hacer esto, es necesario hacer llegar la información del bloque de procesado de imagen (ver Figura 2.2—5) al bloque estimador de estados:

- Señal de detección: '1' cuando se detecte marca válida y '0' cuando no se detecte.
- Posición de la marca: centroide de la marca detectada.

Adicionalmente, es necesario disponer en el estimador una señal de *hover*. Esta señal es de tipo booleana e indica cuando el dron se encuentra en una posición estable, estática y totalmente horizontal con el plano del suelo. La señal de *hover* se activa (hover='1') cuando se cumplen las siguientes condiciones previamente fijadas:

- Las velocidades en todos sus ejes son cercanas a 0 (-0.03 < dx, dy, dz < 0.03 m/s).
- Los ángulos pitch y roll son cercanos a 0 (-0.05 < θ, ϕ < 0.05 rad).

Estas condiciones se han considerado como una solución de compromiso entre error cometido en la estimación y número de medidas válidas.

Se diseña el bloque indicado en la Figura 3.3—2 que generará la señal de *hover* a partir de las condiciones anteriormente expuestas.

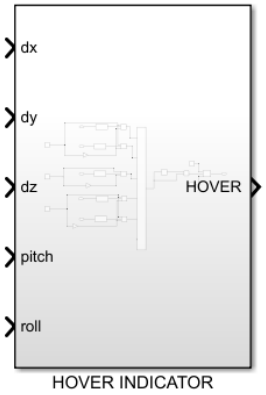


Figura 3.3—2 - Bloque generador de señal de hover

El nuevo estimador de posición se configura a partir del bloque *Kalman Filter* de la librería de Simulink[4].

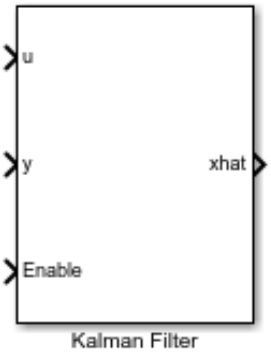


Figura 3.3—3 - Bloque de Filtro de Kalman utilizado como predictor-corrector de posición.

En la Figura 3.3—4 se indica el proceso de configuración del bloque y a continuación se describen los datos introducidos:

[4] Kalman Filter: https://es.mathworks.com/help/ident/ref/kalmanfilter.html

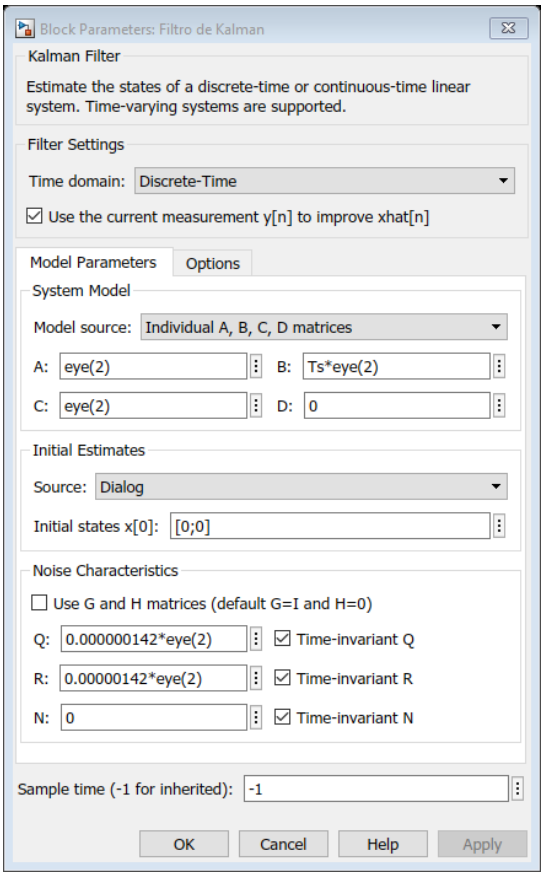


Figura 3.3—4 – Configuración utilizada en el bloque Filtro de Kalman

Filter Settings:

- Selección de dominio tiempo discreto (Filtro de Kalman Discreto).
- Selección del uso de la medida y[k] para mejorar la salida $\hat{x}$[k]. La medida y[k] es la información de posición extraída de la cámara (detección de marcas) como se muestra en la Figura 3.3—5.

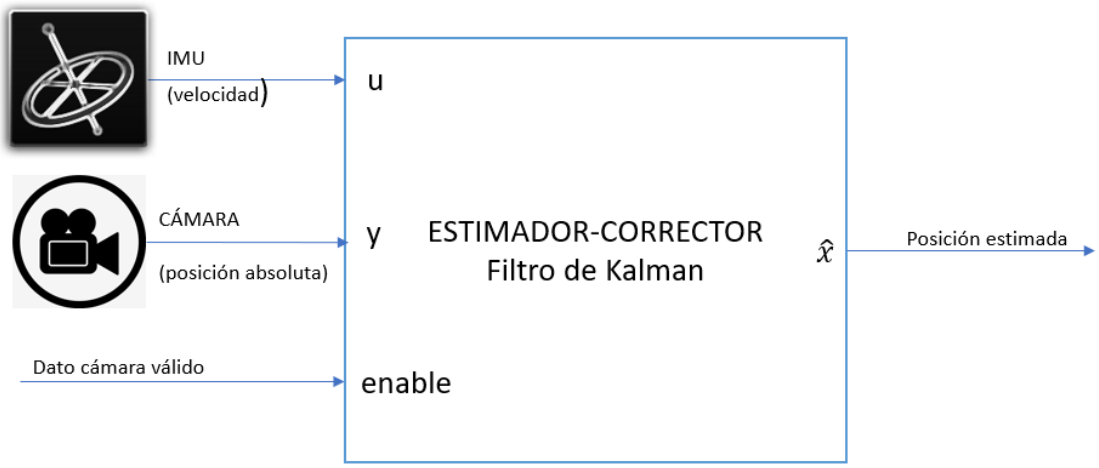


Figura 3.3—5 – Esquema de predictor-corrector. Filtro de Kalman

System Model:

- El modelo se corresponde a un integrador. Integrando la información de velocidad estimada, se obtiene la posición estimada:

$$x_k = x_{k-1} + Ts \cdot u_{k-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} T_s & 0 \\ 0 & T_s \end{pmatrix} \begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix}$$

$$y_k = x_k = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix}$$

Siendo el vector de estado/posición $x_k = \begin{pmatrix} X \\ Y \end{pmatrix}$ y su derivada con respecto al tiempo: el vector de velocidad $x_k = \begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix}$.

Se introducen las matrices del modelo en el bloque:

$$G = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad H = \begin{pmatrix} T_s & 0 \\ 0 & T_s \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad D = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Y también el vector de estados iniciales, que se corresponde con las coordenadas de origen (punto de despegue):

$$x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

El último paso es el cálculo de las matrices de covarianza de ruido del proceso Q y covarianza de medida R.

- Cálculo de la covarianza del ruido de medida. Como se observa en la Figura 3.3—5, las medidas se toman tanto de la IMU como de la cámara. Para medir el ruido, se han tomado medidas con cada uno de estos sensores con el dron en estático como se muestra en la Figura 3.3—6.

Figura 3.3—6 – Ensayo realizado para la caracterización de ruido de medida

En la Figura 3.3—7 se muestra la imagen captada por la cámara. Las marcas detectadas se muestran resaltadas con un rectángulo de color negro.

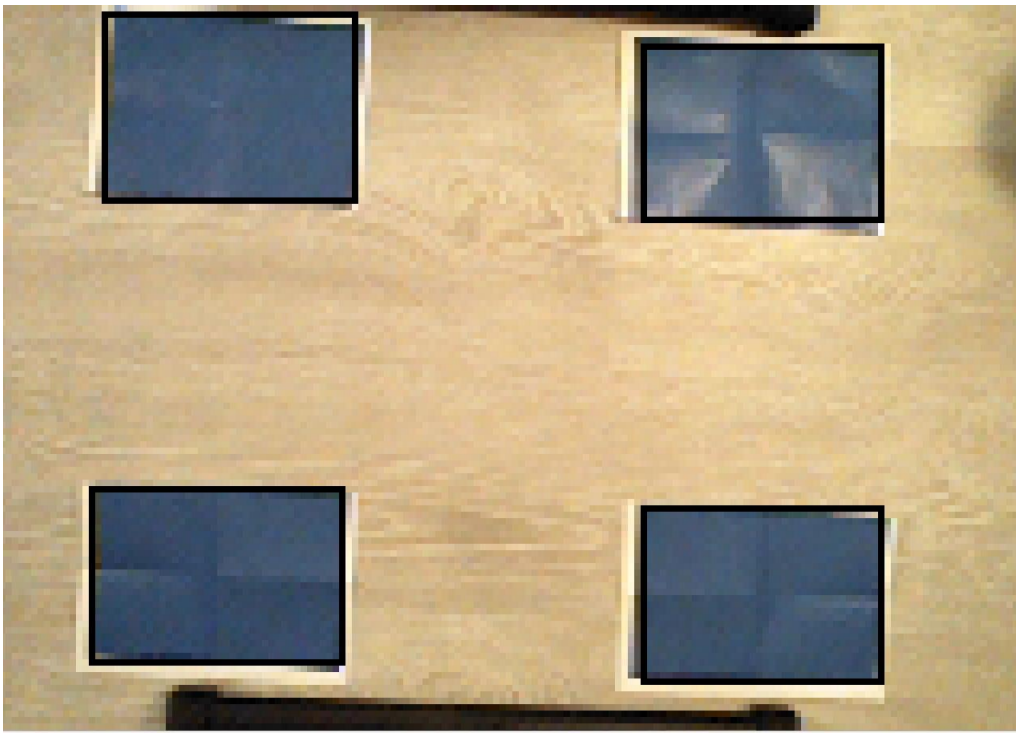


Figura 3.3—7 - Imagen capturada por la cámara del dron durante el ensayo realizado para la caracterización de ruido de medida

En la Figura 3.3—8 se muestran las mediciones de una de las marcas obtenida con la cámara. En la parte superior la coordenada X del dron (frontal) y en la parte inferior la coordenada Y (lateral). Las medidas en el eje de abscisas se representan en metros.

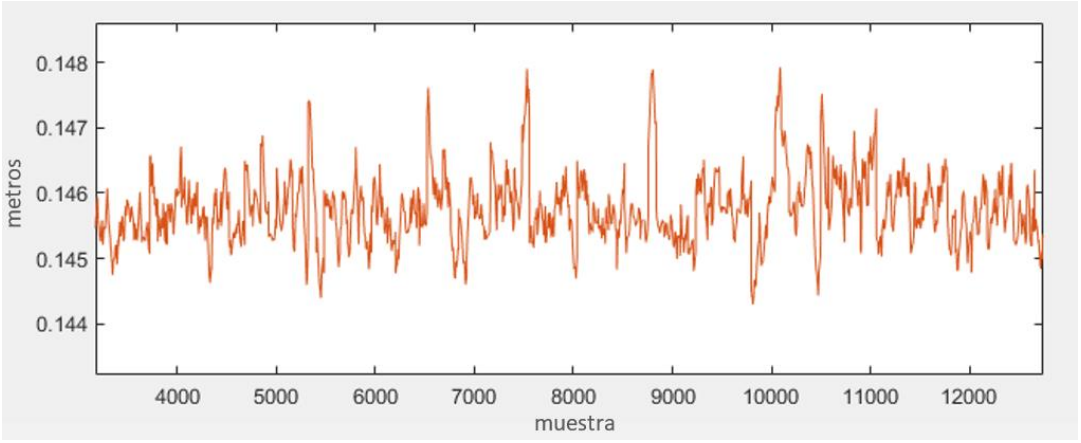


Figura 3.3—8 - Medida de posición relativa obtenida del ensayo realizado para la caracterización de ruido de medida

Se calcula la covarianza del ruido de medida de cada una de las marcas y se extrae la mayor de ellas (peor caso). En el Anexo 1 se muestra el script utilizado para el cálculo de la covarianza del ruido de medida de la cámara.

El valor máximo de covarianza obtenido es el de $1.42 \cdot 10^{-6}$ m^2.

La matriz de covarianza del ruido de la medición R sería la siguiente:

$$R = \begin{pmatrix} 1.42 \cdot 10^{-6} & 0 \\ 0 & 1.42 \cdot 10^{-6} \end{pmatrix}$$

Para la matriz de covarianza de ruido del proceso se selecciona una matriz que cumpla la relación Q < R. Es este caso se selecciona la relación Q = R/10.

$$Q = \begin{pmatrix} 1.42 \cdot 10^{-7} & 0 \\ 0 & 1.42 \cdot 10^{-7} \end{pmatrix}$$

3.4. Resultados simulados y experimentales

En este apartado, se muestran los resultados de las pruebas realizadas con los controladores y estimadores diseñados. Las pruebas de dividen en dos fases: simulación y experimentación. Las simulaciones se realizan utilizando el entorno de simulación proporcionado por Parrot, en el que se introducirán unas marcas en el suelo que sirven al dron para corregir el error de posicionamiento. Una vez verificados los resultados en simulación, se realizan las pruebas experimentales en las que se utilizarán marcas reales situadas en el suelo y el código se ejecutará en tiempo real en el dron.

3.4.1.Pruebas en simulación. Seguimiento de trayectoria.

A continuación, se describe el entorno de simulación y se muestran las pruebas en simulación realizadas, en las que el dron sigue una trayectoria cuadrada para mostrar el efecto de la corrección de posición que se produce al detectar las marcas en el suelo con la cámara vertical del dron.

3.4.1.1. Introducción al entorno de simulación

Durante la realización del trabajo, se ha trabajado con los subsistemas *Simulation Model* y *Flight Visualization* (ver apartado 2.2.1.2). Estos subsistemas proporcionados por Parrot permiten simular vuelos y visualizar los resultados en tiempo real. En la Figura 3.4—1 se muestra un esquema de los bloques mencionados.

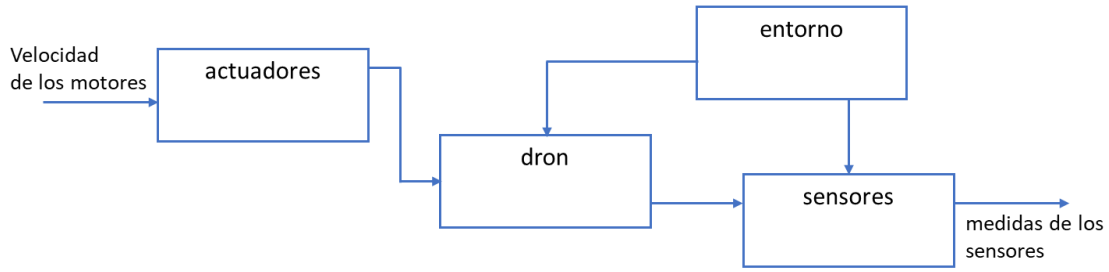


Figura 3.4—1 - Modelo del conjunto dron-entorno.

El subsistema *Simulation Model* contiene 3 bloques:

- Modelo del dron (Multicopter Model): Este bloque, mostrado en la Figura 3.4—2, traduce las señales de actuación de los rotores en las señales de posición, velocidad y aceleración que se generan en el dron según el modelo proporcionado por Parrot. Este

bloque está basado en el bloque 6DOF (Quaternion)[5] perteneciente a la librería Aerospace Blockset de Simulink.

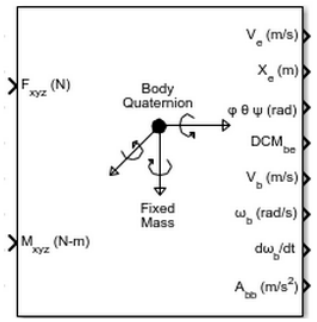


Figura 3.4—2 - Bloque 6DOF (Quaternion)

- Modelo del entorno (Environment Model): Este bloque simula las condiciones de gravedad, temperatura, velocidad del sonido, densidad del aire y campo magnético. Debido a que las ejecuciones se realizan en interior se utiliza un valor constante para cada una de las condiciones como se observa en la Figura 3.4—3.

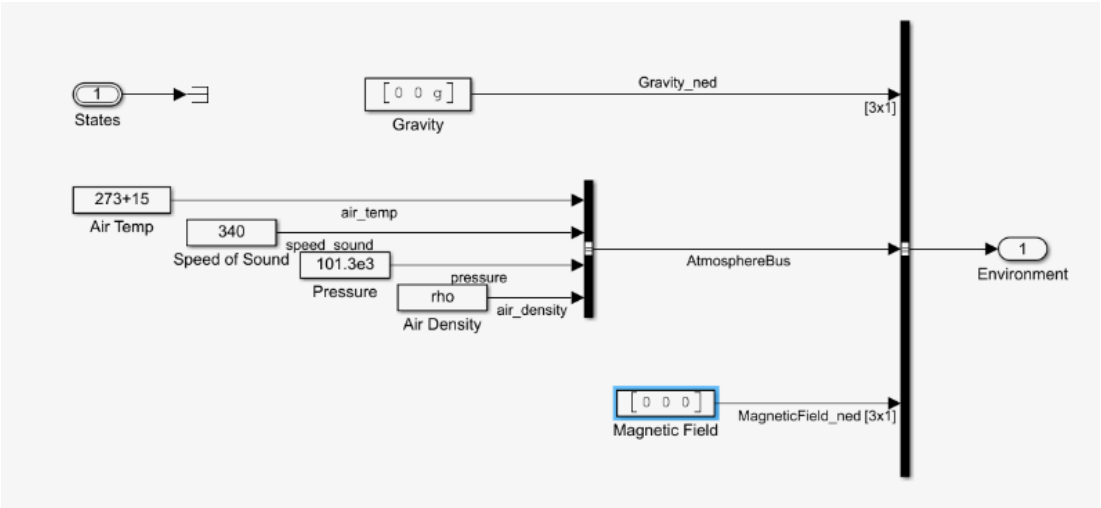


Figura 3.4—3 - Condiciones atmosféricas del modelo

- Modelo de los sensores (Sensor Model): Este bloque proporciona el valor de sensores (IMU, sónar, barómetro) y también proporciona la imagen captada por la cámara. Esta imagen es generada a partir de la animación generada por el subsistema *Flight Visualization*.

[5] https://es.mathworks.com/help/aeroblks/6dofquaternion.html

La librería de Parrot también incluye el subsistema *Flight Visualization* (ver apartado 2.2.1.2), que es el encargado de generar una animación que permite por un lado simular de forma visual el comportamiento del dron y, por otro lado y más importante, obtener una imagen de la cámara integrada del dron que permita simular también la parte de visión artificial del trabajo.

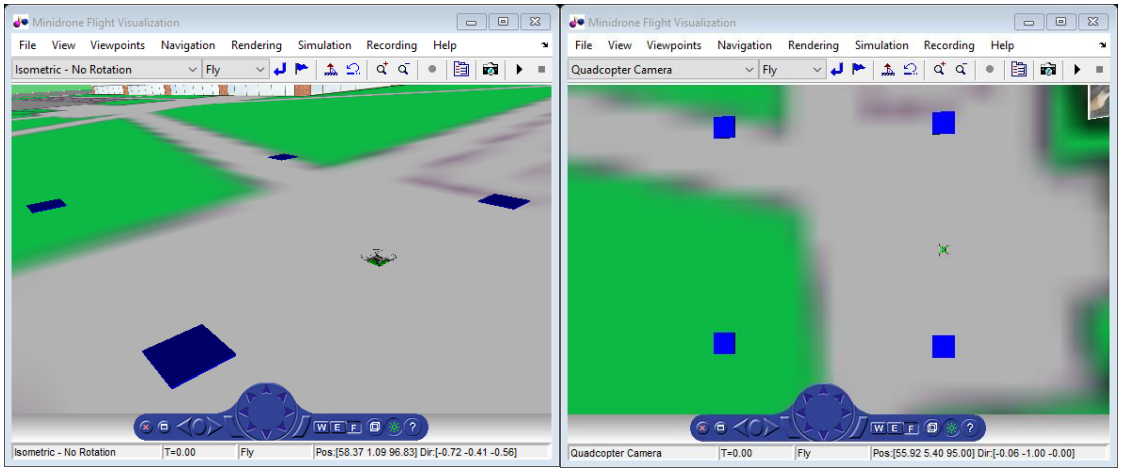


Figura 3.4—4 - Simulación visual de dron y entorno.

Como se observa en la Figura 3.4—4, se ha modificado el entorno proporcionado por Parrot, de forma que se han introducido 4 marcas azules en el plano del suelo, que servirán para que el dron corrija su posición estimada cuando detecte cada una de ellas. Estando el dron en el origen de coordenadas (0 ; 0), las marcas están situadas en las siguientes coordenadas:

(0 ; 1.3)
(-3.5 ; 1.3)
(-3.5 ; -1.7)
(0 ; -1.7)

Al realizar pruebas en simulación con el modelo de dron proporcionado por Parrot, no existe error entre la posición estimada del dron y la posición real. Para forzar un error de posición, las posiciones de las marcas almacenadas en el dron no coincidirán con su posición real, siendo las siguientes:

(0 ; 1.5)
(-3.4 ; 1.5)
(-3.4 ; -2)
(0 ; -2)

De esta forma se fuerza la necesidad de corregir su posición estimada, utilizando como referencia la posición de las marcas detectadas.

3.4.1.2. Resultados de la simulación

A continuación, se indican los resultados de las simulaciones. A la señal obtenida de la cámara, se le ha introducido el ruido de medida caracterizado de forma experimental (ver apartado 3.3.2). Para ello, se utiliza el bloque Band-Limited White-Noise, en el que configura el parámetro *Noise power* como el valor de la covarianza del ruido por el periodo de muestreo como se muestra en la Figura 3.4—5.

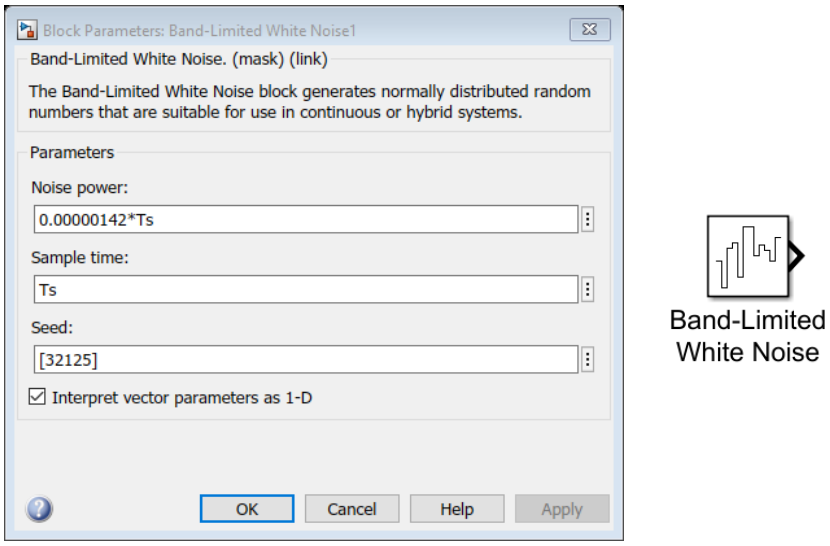


Figura 3.4—5 – Generación de ruido de medida de la cámara en simulación

Antes de comenzar la simulación, se comprueba con el comando *cov* de Matlab que el ruido generado tiene exactamente la covarianza que se ha medido experimentalmente como se describe en el apartado 3.3.2.

```
>> cov(ruido_generado)
    ans =
      1.4042e-06
```

Pruebas sin corrección de posición (lazo abierto)

En la Figura 3.4—6 se muestra el resultado de una simulación en la que la señal de la cámara no corrige la posición calculada a partir de la información de la IMU. Para que la posición no se corrija, se ha forzado a '0' la señal de *enable* del filtro de Kalman (ver Figura 3.3—5). De esta forma, a la salida del Filtro de Kalman 'xhat' se obtiene la integral de la velocidad estimada a partir de la información de la IMU, y en ningún momento se corregirá con la información de posición calculada a partir de la información de la cámara.

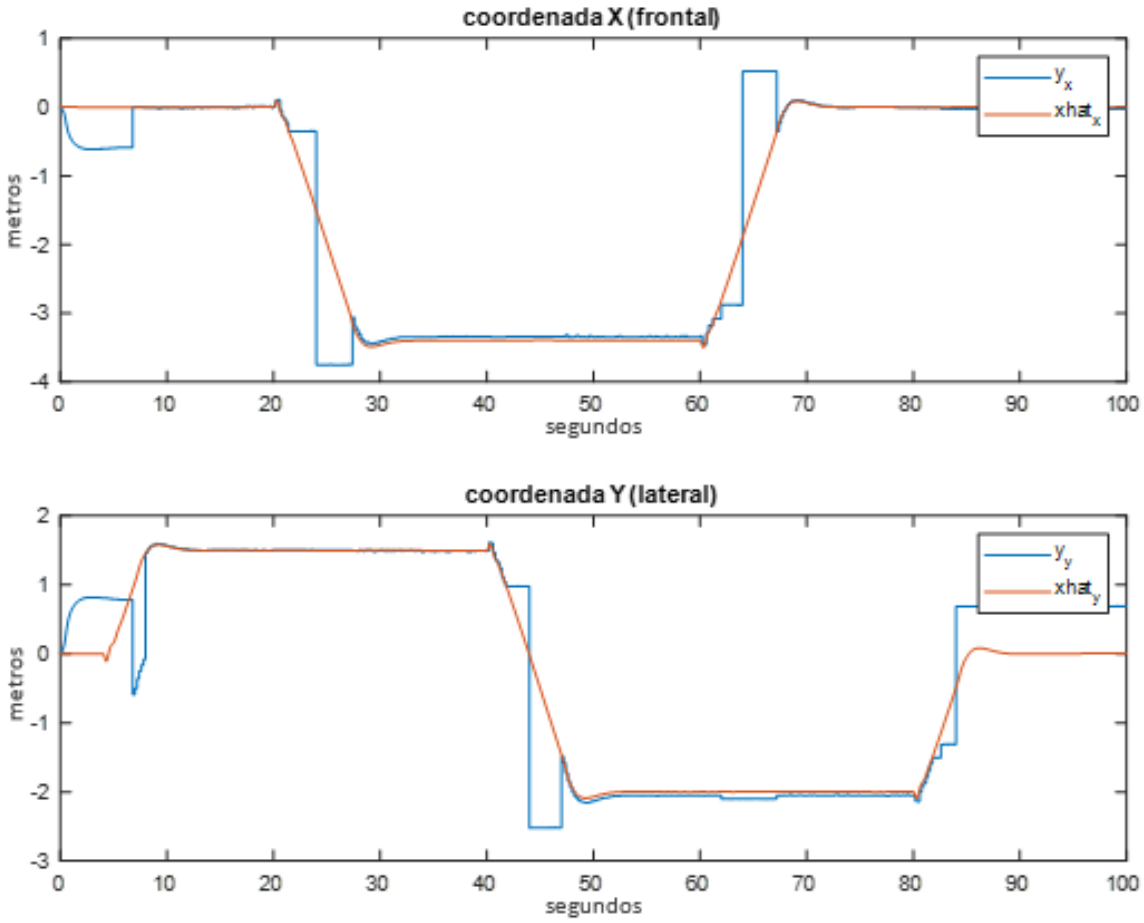


Figura 3.4—6 – Vista general de los resultados de simulación en lazo abierto.

En rojo se representa la posición del dron calculada a partir de la información de la IMU, mientras que en azul se representa la posición del dron calculada a partir de la información de la cámara.

En la Figura 3.4—7 se muestra la coordenada X del dron ampliada, donde se observa que la posición del dron calculada a través de la información de la cámara no coincide con la posición calculada a partir de la información de la IMU.

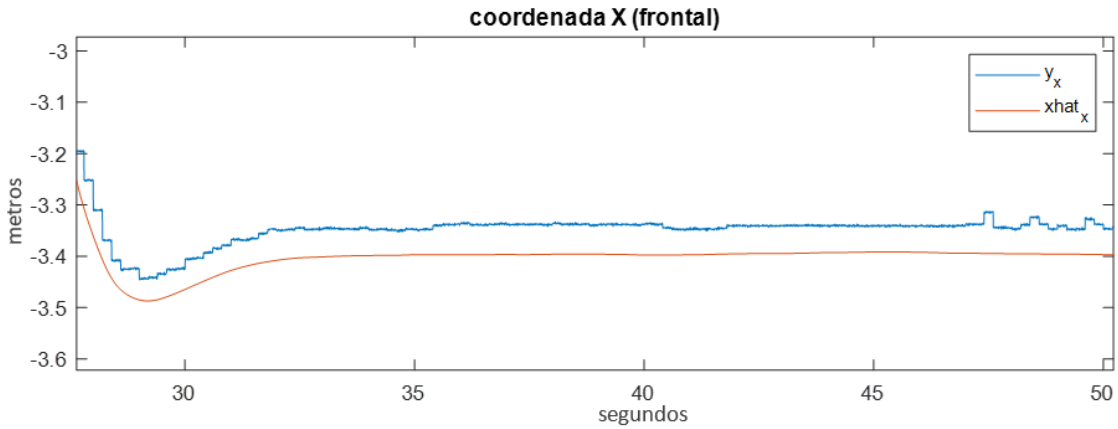


Figura 3.4—7 - Vista en detalle de señal en lazo abierto. Discrepancia entre estimación de posición con IMU con información de la cámara.

Pruebas con corrección de posición (lazo cerrado)

En la Figura 3.4—8 se muestra el resultado de la misma simulación realizada en el apartado anterior, pero en este caso la posición calculada a partir de la información de la cámara sí se utiliza para corregir la posición calculada a partir de la información de la IMU.

Se representan las señales correspondientes a los puertos '$\hat{x}$', '*y*' y '*enable*' del filtro de Kalman (ver Figura 3.3—5).

- En azul se representa la salida de la planta 'y', que es la entrada del filtro de Kalman, que es la posición del dron calculada a partir de la información de la cámara.
- Las aspas negras señalan cuándo la información de la cámara es válida, es decir, cuándo está activa la señal de entrada 'enable' del filtro de Kalman.
- En rojo se representa la salida del filtro de Kalman 'xhat' ('$\hat{x}$'), es decir la posición promediada entre la estimada y corregida del dron.

Cuando la entrada 'enable' no está activa, la salida '$\hat{x}$' integra la velocidad estimada a partir de la información de la IMU. En el instante en el que se activa la señal 'enable' del filtro de Kalman, esto es cuando la imagen de la cámara es válida, se corregirá la posición estimada con la entrada 'y', que es la información de posición calculada a partir de la información de la cámara.

La imagen se considera válida (y por tanto se activa la señal enable) cuando se detecta una marca en el suelo durante un estado de hover, es decir, cuando el dron está parado y paralelo al suelo.

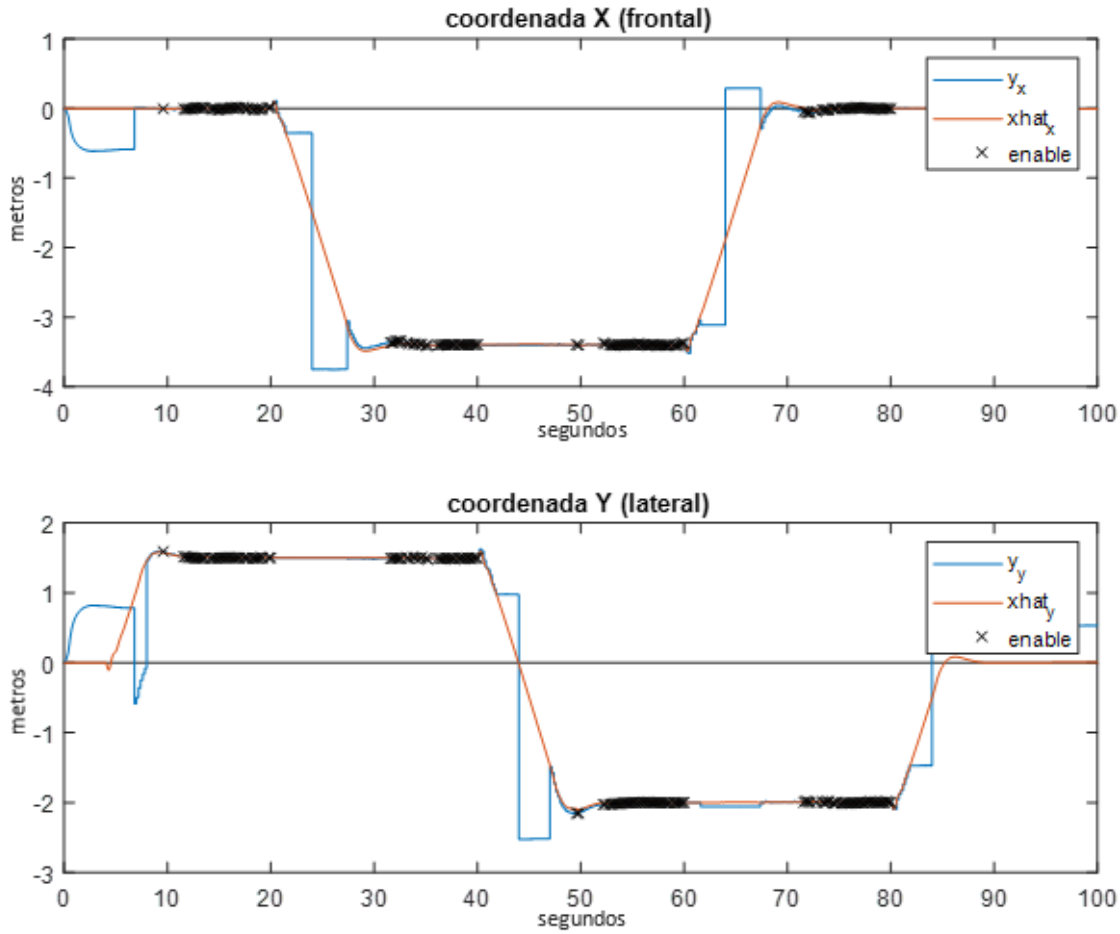


Figura 3.4—8 - Vista general de los resultados de la simulación en lazo cerrado. La posición estimada se corrige cuando se detecta la marca (enable)

Figura 3.4—9 se muestra ampliada la coordenada X del dron en un intervalo de tiempo en el que se realiza una de las correcciones. Se observa que cada vez que se dispone de imagen válida de la cámara, la posición calculada a partir de la información de la IMU (rojo) se corrige. Tras varias correcciones, la posición calculada a partir de la IMU y la posición calculada a partir de la información de la cámara coinciden.

Se diferencian 4 periodos de tiempo en los que se activa la señal *enable*: entre t=10 hasta t=20, entre t=30 hasta t=40, entre t=50 hasta t=60 y entre t=70 hasta t=80. En estos periodos de tiempo el dron se encuentra en estado hover y se ha captado una de las marcas azules del suelo. En cada uno de los periodos de activación de la señal *enable* el dron se encuentra sobre una de las cuatro marcas situadas en los vértices del cuadrado que forma la trayectoria del dron.

En la Figura 3.4—9 se muestra ampliada las señales correspondientes a la coordenada X (eje frontal del dron). Se observa que entre los instantes t=31 y t=34, se han realizado varias correcciones hasta que la salida del Filtro de Kalman (señal roja) coincide con la información de

posición calculada a partir de la información de la cámara. A partir de este instante, las activaciones de la señal *enable* del filtro de Kalman al detectar de nuevo la misma marca, producen correcciones sin efecto sobre la posición estimada puesto que coinciden ambas informaciones.

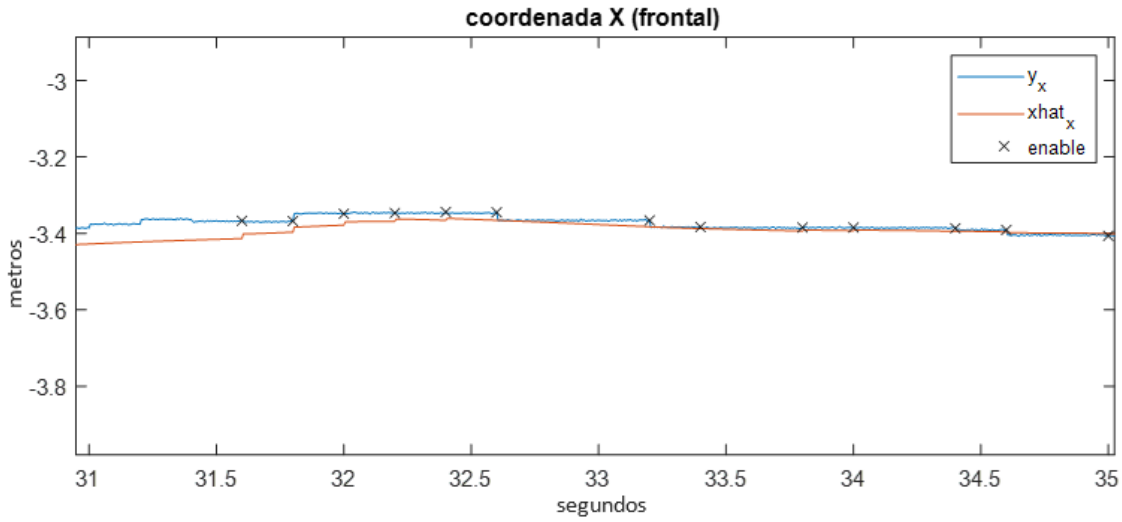


Figura 3.4—9 - Vista en detalle de corrección de posición en simulación. Zoom entre instantes 31 a 35 segundos.

Se observa en los resultados simulados que la discrepancia entra las posiciones calculadas por IMU y cámara desaparece al detectar una de las marcas del suelo y, por tanto, el error de posición no se acumula durante la simulación.

3.4.2.Pruebas experimentales: Seguimiento de trayectoria

A continuación, se muestra una prueba en ejecución con dron real. En la Figura 3.4—10 se muestra el entorno preparado para la prueba. En este caso se trata de una trayectoria en línea recta en la que el dron despega sobre una de las marcas azules con el objetivo de que se posicione sobre ella, corrigiendo el posible error de posición generado durante la maniobra de despegue. A continuación, el dron avanza hasta llegar a la segunda marca azul donde volverá a corregir su posición y, por último, volverá a la marca inicial. Debido a que el color del suelo utilizado en la prueba es bastante homogéneo, ha sido necesario colocar marcas de color marrón para generar un patrón irregular en el suelo para disponer de suficiente contraste de color. Esto es necesario para el correcto funcionamiento del hardware de detección de Flujo óptico (Optical Flow) descrito en el apartado 2.2, no influyendo al algoritmo de detección de las marcas azules. La posición de las marcas marrones es aleatoria y no es necesario conocer las coordenadas de su posición.

Figura 3.4—10 - Escenario utilizado para las pruebas de ejecución con trayectoria de avance retroceso en línea recta.

En la Figura 3.4—11 se muestra el resultado de la ejecución. Para ello, se representan las señales correspondientes a los puertos '$\hat{x}$', '*y*' y '*enable*' del filtro de Kalman (ver Figura 3.3—5). Al igual que en los resultados de las pruebas de simulación en lazo cerrado, se representa en azul la posición del dron calculada a partir de la información de la cámara ('y'), en rojo la promediada entre la estimada y corregida del dron '$\hat{x}$' y se señalan con aspas negras los instantes en los que la marca es detectada con imagen válida y, por lo tanto, se realiza corrección de posición ('enable').

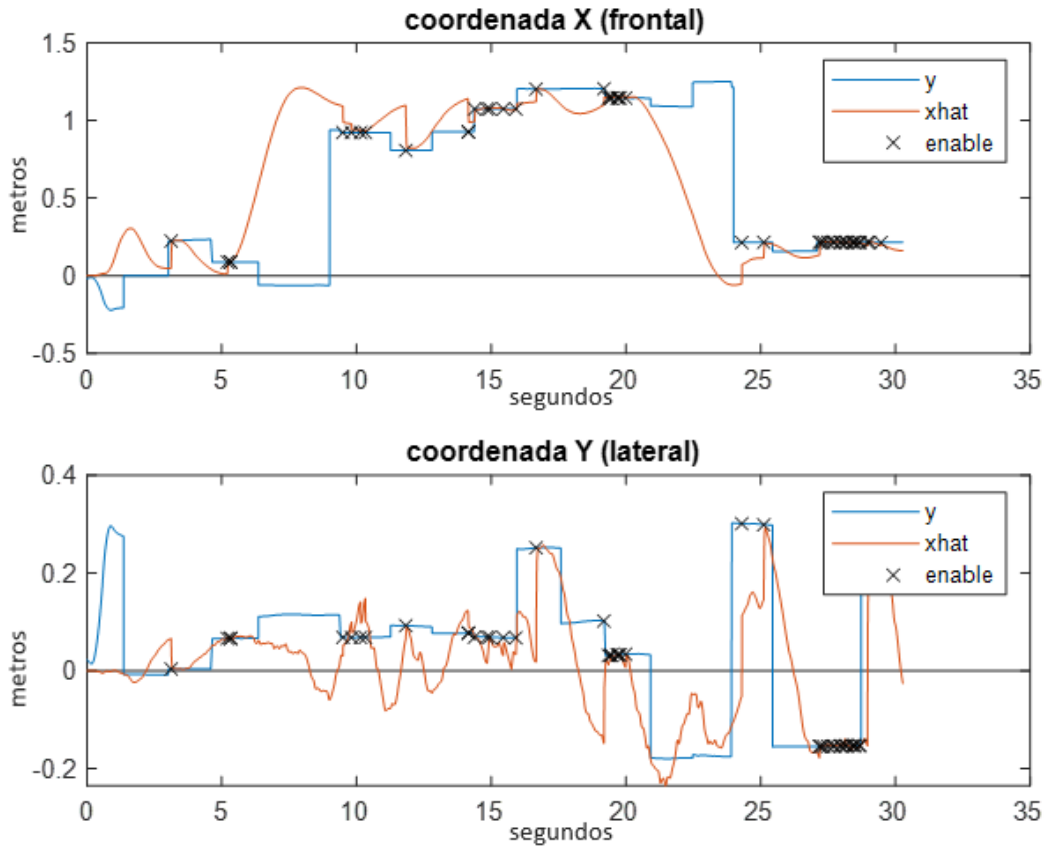


Figura 3.4—11 – Vista general de los resultados obtenidos en las pruebas experimentales

Durante los primeros 3 segundos de ejecución, el dron está despegando del suelo. En este periodo de tiempo, la información captada por la cámara no es válida por estar el dron en movimiento y por eso en este periodo no hay ninguna corrección por el Filtro de Kalman.

En la Figura 3.4—12 se muestra que pasado el segundo 3 se detecta la primera corrección de posición, de unos 15 cm en coordenada X y unos 6 cm en coordenada Y.

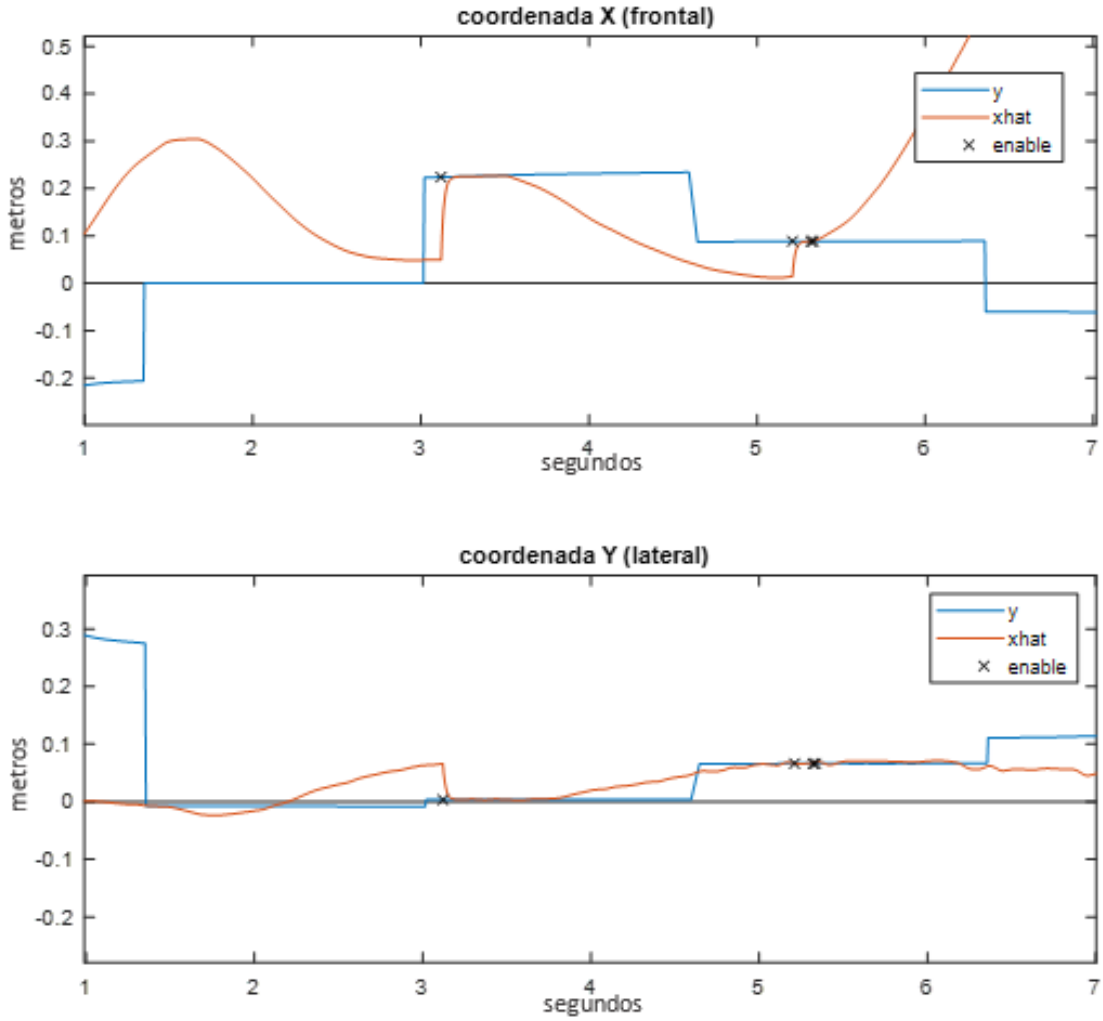


Figura 3.4—12 – Vista en detalle de las pruebas experimentales entre instantes 1 y 7 segundos: Despegue y primeras correcciones de posición.

En la Figura 3.4—13 se muestra las correcciones realizadas sobre la segunda de las marcas. En este caso, el dron se ha desplazado a la posición (1, 0), en la cual se encuentra la segunda marca y será la posición con la que el dron corregirá su posición estimada. Entre los instantes 9 y 11 se observa que se realizan 4 correcciones de posición, estas correcciones se realizan una vez que el dron se ha detenido en la coordenada (1, 0) en posición de *hover*, es decir estable con velocidad casi nula.

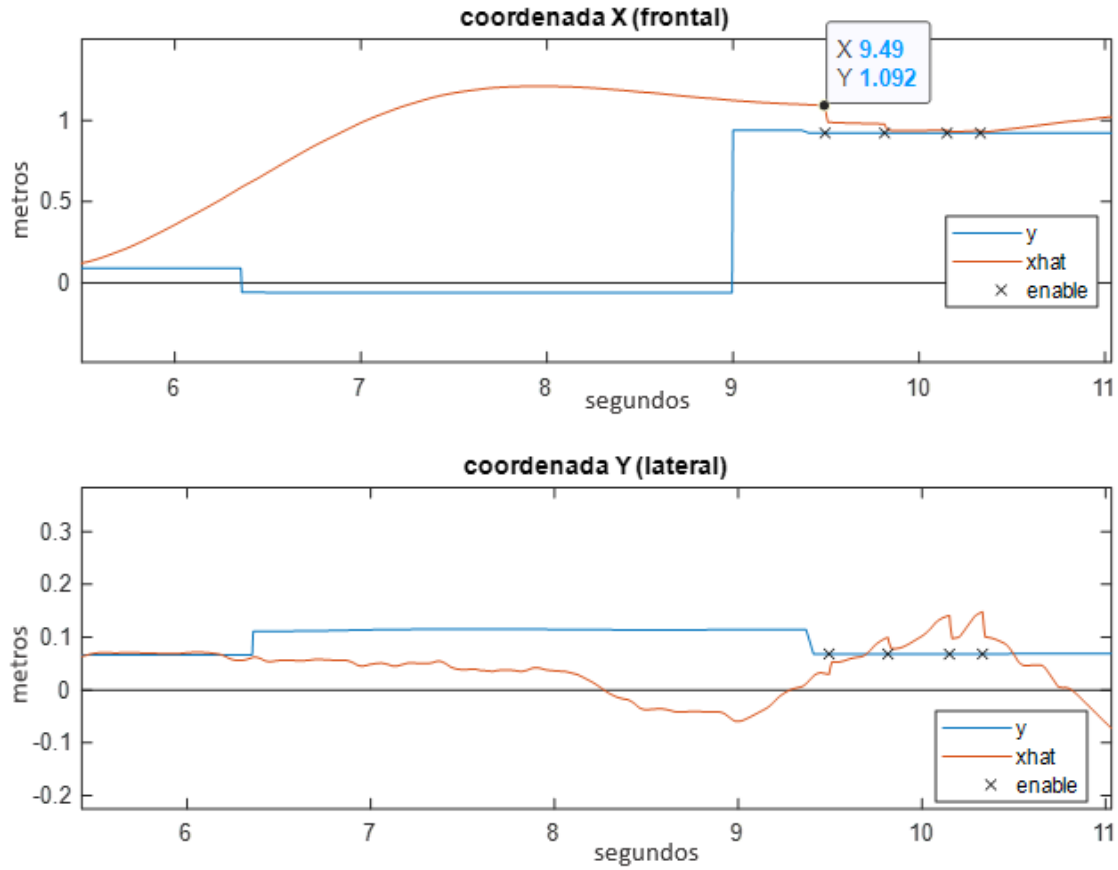


Figura 3.4—13 - Vista en detalle de las pruebas experimentales entre instantes 6 y 11 segundos: Correcciones de posición en segunda marca

En la Figura 3.4—14 se muestra ampliada la gráfica correspondiente a la coordenada X durante el periodo de tiempo comprendido ente el segundo 10 y 22 de ejecución. Se observa que se realizan varias correcciones cuando se detecta la marca azul del suelo. En los instantes 12 y 14 aproximadamente se producen grandes correcciones de unos 25 y 10 cm respectivamente, lo que implica que el control del dron actúa para seguir a la señal de referencia, que en estos instantes le macaría la posición X=1 metro. Entre los instantes 14 y 16, las correcciones producidas son prácticamente nulas, puesto que en ese periodo la posición calculada a partir de la información de la cámara coincide con la posición calculada con la información de la IMU.

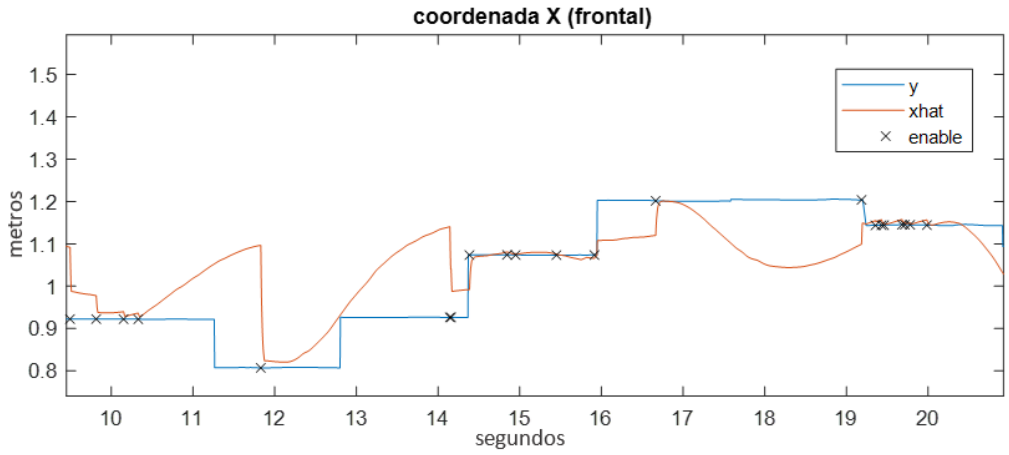


Figura 3.4—14 - Vista en detalle de las pruebas experimentales entre instantes 10 y 20 segundos: Coordenada X. Correcciones en segunda marca.

En la Figura 3.4—15 se muestran las últimas correcciones de la trayectoria. A partir de instante 20, el dron vuelve a la posición de partida, donde se encuentra la marca inicial. Entre los segundos 20 y 24 el dron se encuentra en movimiento y la imagen de la cámara no es válida. A partir del segundo 24 el dron se encuentra sobre la marca inicial y a partir de ese instante es cuando se detectan estados de *hover* en los que la información de posición calculada a partir de la información de la cámara es válida.

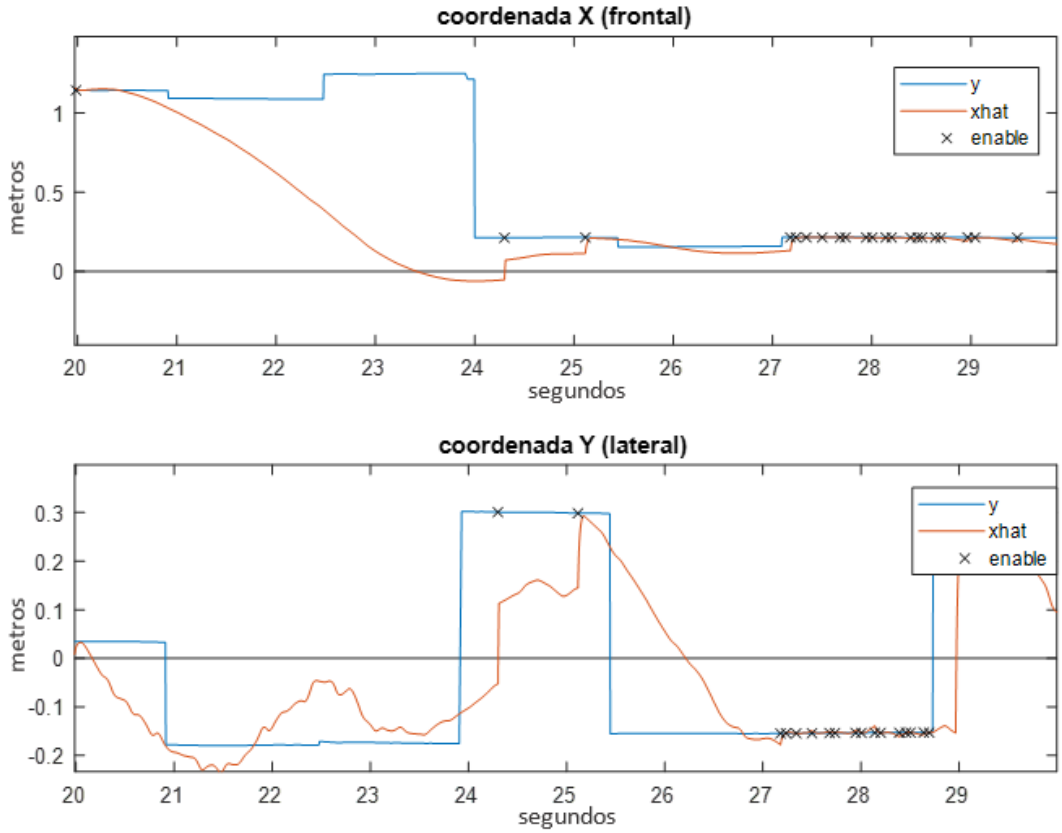


Figura 3.4—15 - Vista en detalle de las pruebas experimentales entre instantes 20 y 30 segundos: Correcciones realizadas en la última posición

4. CONCLUSIONES

El presente trabajo aborda la tarea de desarrollar un sistema de navegación y posicionamiento de drones para su uso en seguimiento de trayectorias en entornos interiores. Para realizar la corrección del error acumulado por los sensores inerciales se utiliza la cámara situada en la parte inferior del minidrón Parrot Mambo y un conjunto de marcas en el suelo con posición conocida, a modo de balizas pasivas. El punto de partida es la librería *Simulink Support Package for Parrot Minidrones* proporcionada por MathWorks y Parrot.

Se ha observado que, al estar la cámara integrada en el dron, todo movimiento u oscilación que realice el dron afecta la exactitud de la posición que se obtiene de la imagen. Por esto, ha sido necesario modificar la respuesta del control proporcionado en el modelo inicial para suavizar los movimientos de aproximación al punto objetivo de la trayectoria. Además, se observa que, para un correcto funcionamiento del sistema de posicionamiento mediante balizas pasivas, es necesario introducir en la trayectoria a seguir puntos de espera o "wait points". En estos puntos se encontrarán las marcar utilizadas para corregir la posición estimada del dron. Para que la corrección sea precisa es necesario que el dron se encuentre en posición de "hover", es decir, con movimientos nulos o casi nulos, totalmente paralelo al suelo y con visión completa de la marca.

En comparación con sistemas basados en sensores externos al dron, el sistema implementado elimina la problemática que presenta la aparición de obstáculos entre en sistema de posicionamiento externo y el dron. Adicionalmente, al realizarse todo el mecanismo de obtención de posición de forma local se eliminan los problemas que resultan de utilizar un canal de comunicaciones para cerrar el lazo de control.

En lo que respecta a la solución de visión artificial, inicialmente se ha trabajado en una solución de marcas basadas en códigos de barras bidimensionales, estas pueden contener información de la trayectoria a seguir como puede ser próximos puntos objetivos. Sin embargo, esta solución no se ha podido implementar debido a que el hardware del dron utilizado no tiene capacidad suficiente para realizar las tareas necesarias de tratamiento de imagen en tiempo real. Como alternativa, se ha estudiado la opción de realizar las tareas de tratamiento de imagen en un PC remoto, enviando mediante conexión bluetooth la imagen captada por la cámara y recibiendo de vuelta la información extraída de la imagen. Durante las pruebas realizadas, se observa que

la capacidad de la conexión es insuficiente generándose un retardo que se va incrementando según avanza la ejecución. Dadas las limitaciones computacionales del PM, se ha optado por una solución basada en marcas sencillas y de un color determinado ubicadas en el suelo a lo largo de la trayectoria.

La información de localización absoluta obtenida de la cámara, y válida cuando el dron se encuentra en posición de "hover", es utilizada para corregir la posición estimada del dron proporcionada por su IMU. Para ello, se ha utilizado un filtro de Kalman, una vez caracterizado experimentalmente el ruido de medida de la cámara.

De las pruebas realizadas en simulación y en experimentación se puede concluir que:

- Las ventajas de tamaño y coste del PM quedan atenuadas por su limitada capacidad de cálculo, especialmente para el procesamiento de imágenes en tiempo real. Por otra parte, la comunicación disponible permite la interacción con un centro remoto siempre que el control sea local.
- Para aprovechar la captura de la cámara integrada en el sistema de posicionamiento se requiere mejorar el sistema de estabilización original del minidrón. Para ello, se han introducido "wait points" a lo largo de la trayectoria actuando como balizas pasivas.
- El sistema de corrección implementado mediante un Filtro de Kalman, a partir de la información de la cámara, es necesario para evitar las derivas acumulativas asociadas a los sensores inerciales embarcados.
- Los modelos proporcionados por la herramienta Matlab/Simulink reducen la curva de aprendizaje de trabajo con el Parrot Mambo y son de gran ayuda en la fase de simulación de algoritmos, pero requieren de ajustes que adecúen los resultados experimentales a los previstos en simulación.

5. TRABAJO FUTURO

El trabajo se ha realizado con un dron de bajo coste orientado a usos domésticos o recreativos, sin embargo, para sacar un mayor aprovechamiento del sistema sensorial embarcado y la estimación predictor-corrector de posicionamiento se requiere de una mayor capacidad de cálculo. Por tanto, cabe retomar algunas de las propuestas planteadas y no implementadas en este TFM si se dispone de versiones del PM con mayor capacidad de procesamiento. A modo de ejemplo se cita la ejecución en tiempo real de algoritmos de tratamiento de imagen con marcas de posicionamiento más complejas.

Entre las limitaciones más relevantes detectadas en la implementación está la captura estable de la cámara, para ello se requiere que el dron se encuentre en posición “hover”. La mejora en los sistemas de estabilización del dron reducirá los tiempos de captura válida y con ello los tiempos de respuesta de los controladores asociados al posicionamiento y seguimiento de trayectorias.

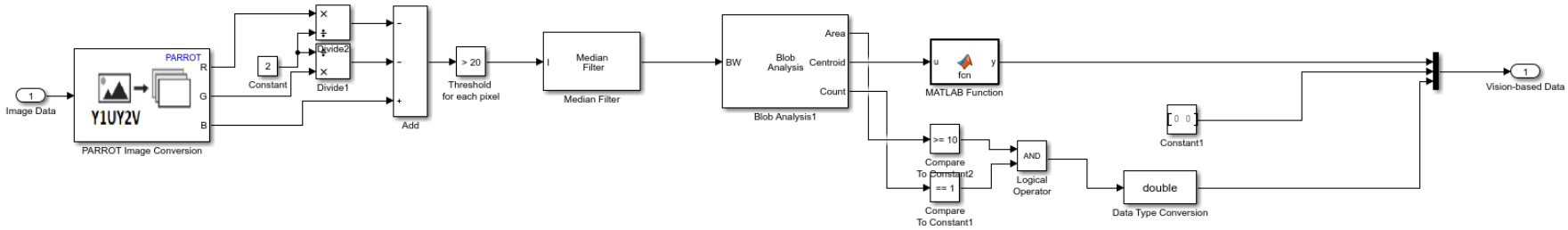
Superados estos nuevos retos, cabe abordar el control de formaciones de varias unidades de PM aprovechando la experiencia previa con robots P3-DX.

ANEXOS

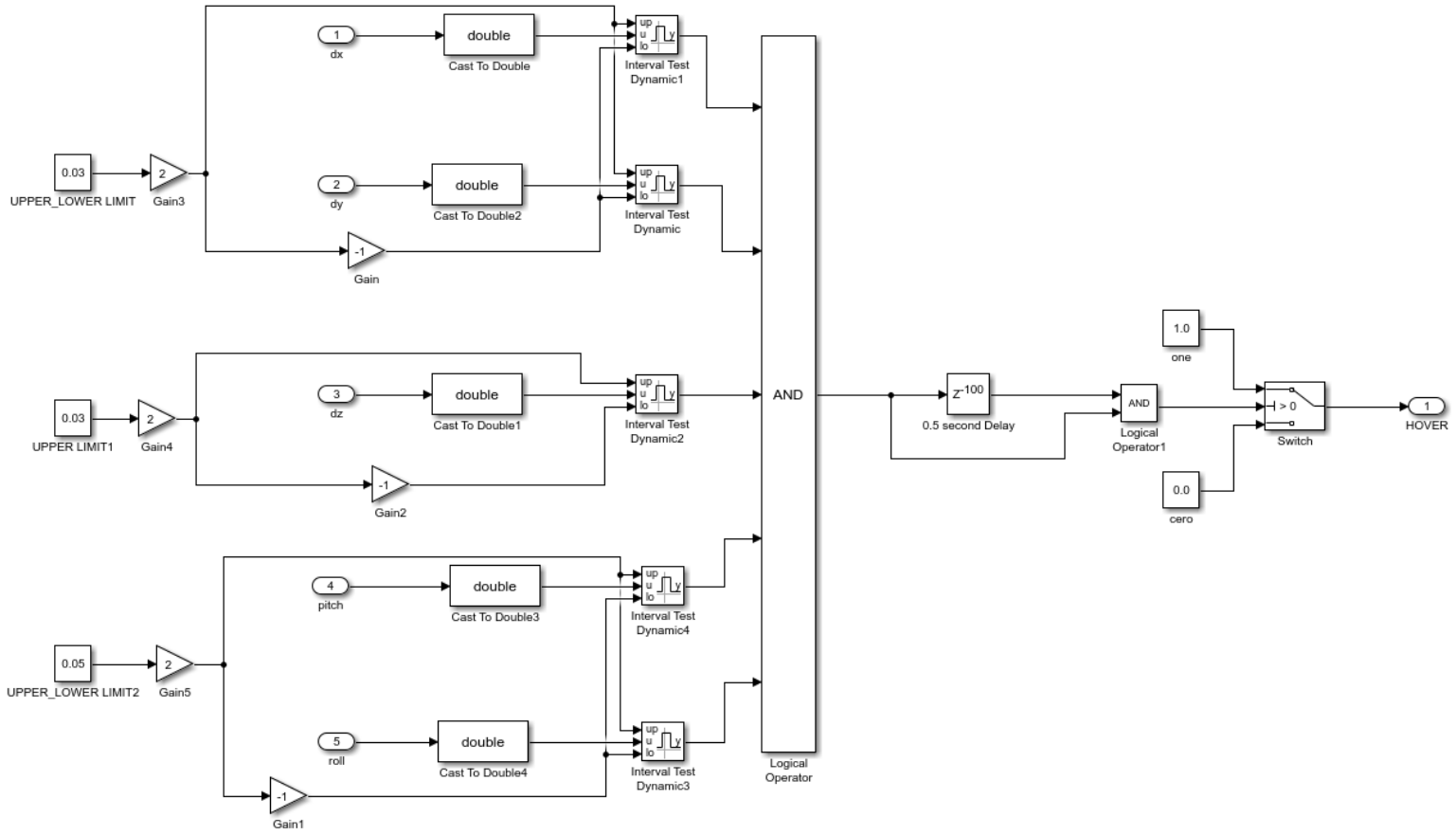
1. SCRIPT DE CÁLCULO DE COVARIANZA DE MEDIDA DE LA CÁMARA

```
load('01_sup_izq.mat')
datos=cc_Btag_pose_absolute(1000:end,:);
figure('Name','01');
plot(datos),hold on;
cov_x(1)=cov(datos(:,1))
cov_y(1)=cov(datos(:,2))

load('02_sup_dch.mat')
datos=cc_Btag_pose_absolute(1000:end,:);
figure('Name','02');
plot(datos),hold on;
cov_x(2)=cov(datos(:,1))
cov_y(2)=cov(datos(:,2))

load('03_inf_dch.mat')
datos=cc_Btag_pose_absolute(1000:end,:);
figure('Name','03');
plot(datos),hold on;
cov_x(3)=cov(datos(:,1))
cov_y(3)=cov(datos(:,2))

load('04_inf_izq.mat')
datos=cc_Btag_pose_absolute(1000:end,:);
figure('Name','04');
plot(datos),hold on;
cov_x(4)=cov(datos(:,1))
cov_y(4)=cov(datos(:,2))

cov_max_x=max(cov_x)
cov_max_y=max(cov_y)

cov_max=max([cov_max_x, cov_max_y])
```

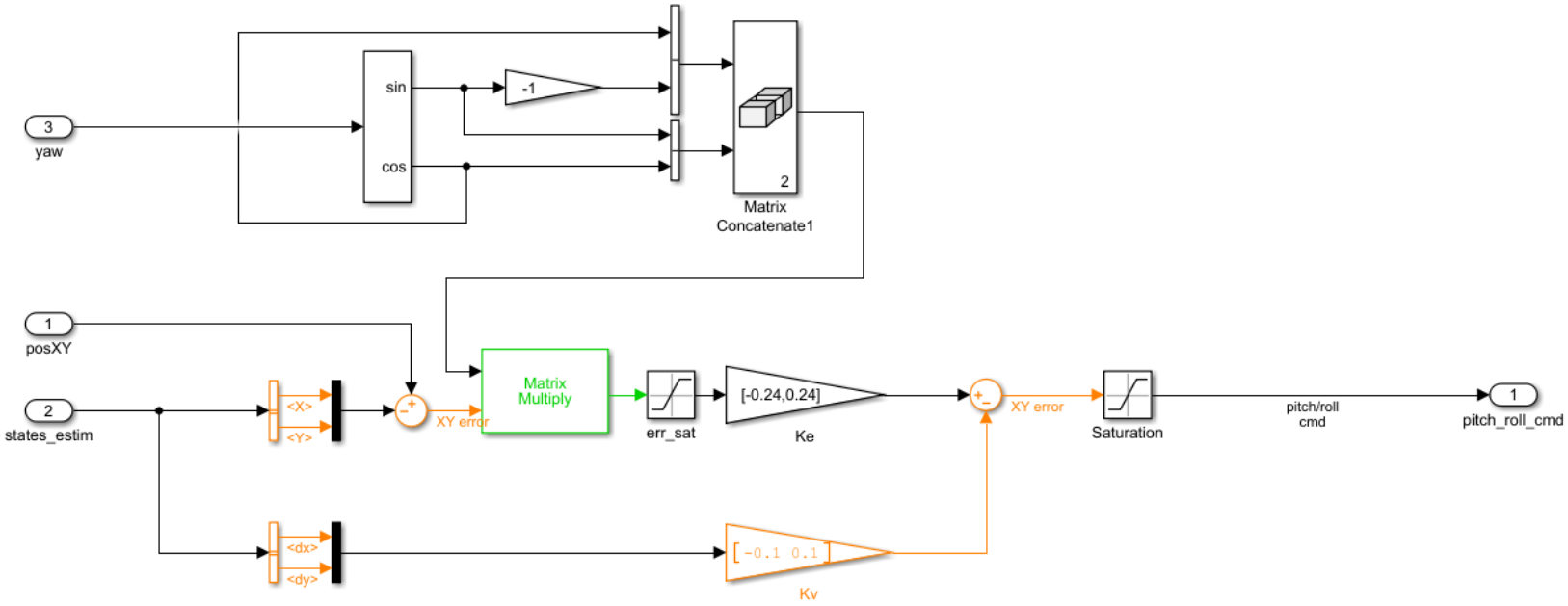
2. SISTEMA DE PROCESADO DE IMAGEN PARA DETECCIÓN DE MARCAS



3. DETECTOR DE ESTADO HOVER



4. CONTROL DE POSICIÓN

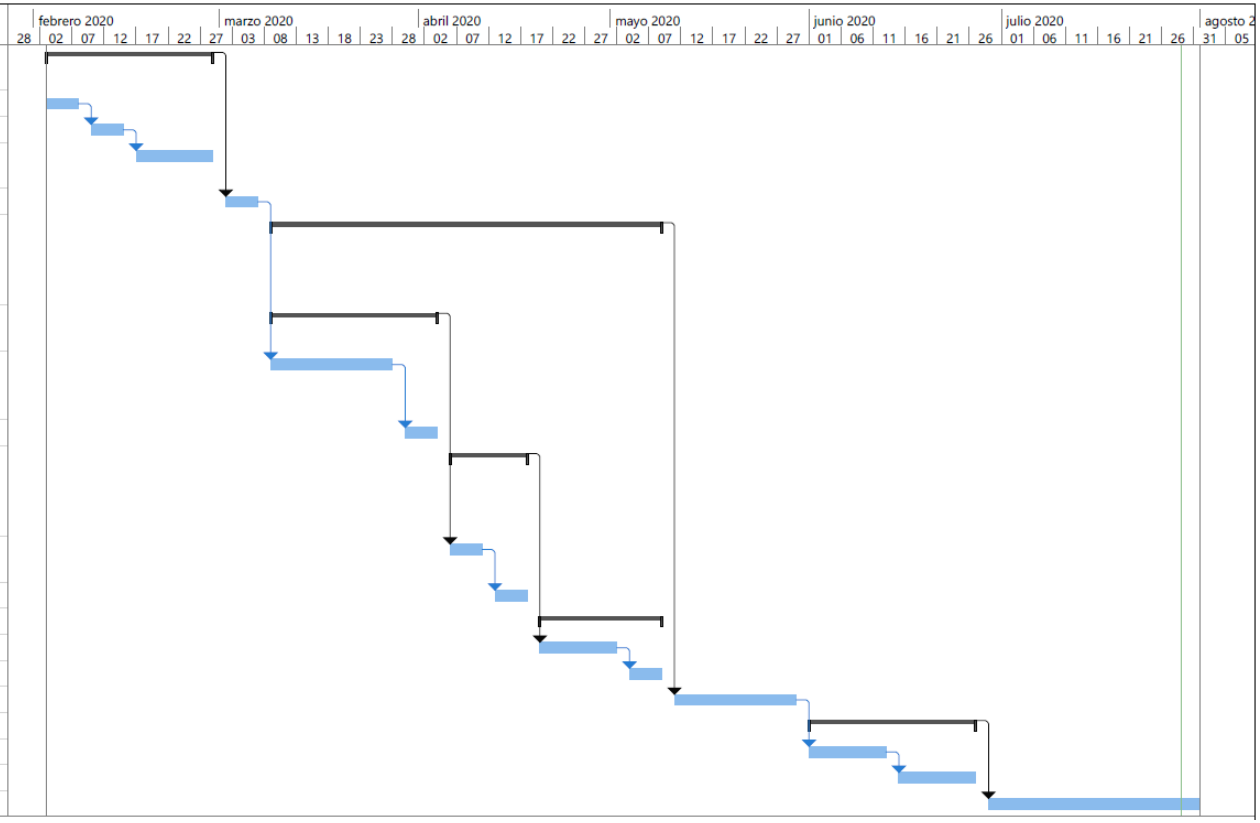


PLANIFICACIÓN Y DIAGRAMA DE GANT

A continuación se muestra la planificación temporal de las fases del proyecto y el diagrama de Gant asociado.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Análisis de software**	**20 días**	**lun 03/02/20**	**vie 28/02/20**
2	Estudio de librerías Simulink involucradas	1 sem	lun 03/02/20	vie 07/02/20
3	Estudio entorno de simulación	1 sem	lun 10/02/20	vie 14/02/20
4	Evaluación modelo inicial	2 sem.	lun 17/02/20	vie 28/02/20
5	Estudio y modificaciones de controlador de posición	1 sem	lun 02/03/20	vie 06/03/20
6	**Desarrollo solución visión artificial**	**45 días**	**lun 09/03/20**	**vie 08/05/20**
7	**Solución basada en reconocimiento y decodificación de códigos BIDI**	**20 días**	**lun 09/03/20**	**vie 03/04/20**
8	Desarrollo algoritmo tratamiento de imagen	3 sem.	lun 09/03/20	vie 27/03/20
9	Pruebas	1 sem	lun 30/03/20	vie 03/04/20
10	**Estudio de viabilidad de PC remoto (comunicación bluetooth)**	**10 días**	**lun 06/04/20**	**vie 17/04/20**
11	Estudio comunicación bluetooth	1 sem	lun 06/04/20	vie 10/04/20
12	Pruebas	1 sem	lun 13/04/20	vie 17/04/20
13	**Solucion basada en reconocimiento de marcas simples**	**15 días**	**lun 20/04/20**	**vie 08/05/20**
14	Desarrollo algoritmo tratamiento de imagen	2 sem.	lun 20/04/20	vie 01/05/20
15	Pruebas	1 sem	lun 04/05/20	vie 08/05/20
16	Solución de predicción-corrección de posición estimada	3 sem.	lun 11/05/20	vie 29/05/20
17	**Pruebas final de desarrollo**	**20 días**	**lun 01/06/20**	**vie 26/06/20**
18	Pruebas en simulación	2 sem.	lun 01/06/20	vie 12/06/20
19	Pruebas en ejecución	2 sem.	lun 15/06/20	vie 26/06/20
20	Redación de la memoria	5 sem.	lun 29/06/20	vie 31/07/20

Id	Nombre de tarea
1	**Análisis de software**
2	Estudio de librerías Simulink involucradas
3	Estudio entorno de simulación
4	Evaluación modelo inicial
5	Estudio y modificaciones de controlador de posición
6	**Desarrollo solución visión artificial**
7	**Solución basada en reconocimiento y decodificación de códigos BIDI**
8	Desarrollo algoritmo tratamiento de imagen
9	Pruebas
10	**Estudio de viabilidad de PC remoto (comunicación bluetooth)**
11	Estudio comunicación bluetooth
12	Pruebas
13	**Solucion basada en reconocimiento de marcas simples**
14	Desarrollo algoritmo tratamiento de imagen
15	Pruebas
16	Solución de predicción-corrección de posición estimada
17	**Pruebas final de desarrollo**
18	Pruebas en simulación
19	Pruebas en ejecución
20	Redación de la memoria



PRESUPUESTO

En esta sección del documento se recoge el presupuesto del proyecto.

1. MANO DE OBRA

A continuación, se muestra las partidas correspondientes a la mano de obra derivada de la ejecución del trabajo:

DESCRIPCIÓN	HORAS	PRECIO/HORA	IMPORTE TOTAL
DESARROLLO (INGENIERÍA)	1000	50 €	50.000 €
REDACCIÓN	160	50 €	8.000 €
		IMPORTE TOTAL	**58.000 €**

2. RECURSOS MATERIALES

A continuación, se muestra las partidas correspondientes a los recursos materiales (hardware y software) utilizados en la realización del trabajo:

DESCRIPCIÓN	UNIDADES	PRECIO UNITARIO	IMPORTE TOTAL
PC	1	900 €	900 €
WINDOWS 10	1	145 €	145 €
MINIDRON PARROT MAMBO	2	100 €	200 €
BATERÍA PARROT	2	20 €	40 €
MATLAB	1	800 €	800 €
SIMULINK	1	1.200 €	1.200 €
AEROSPACE BLOCKSET	1	500 €	500 €
AEROSPACE TOOLBOX	1	460 €	460 €
COMPUTER VISION TOOLBOX	1	500 €	500 €
SIMULINK 3D ANIMATION	1	460 €	460 €
		IMPORTE TOTAL	**5.205 €**

3. IMPORTE TOTAL

A continuación, se indica el importe total del Trabajo Fin de Máster:

DESCRIPCIÓN	IMPORTE
MANO DE OBRA	58000 €
RECURSOS MATERIALES	5205 €
IMPORTE TOTAL	**63.205 €**

BIBLIOGRAFÍA

[1] Katsuhiko Ogata "Sistemas de Control en Tiempo Discreto" Segunda Edición, Sección 8.4, Ed. Prentice Hall.

[2] Sontag, Eduardo (1998), “Mathematical Control Theory: Deterministic Finite Dimensional Systems”. Second Edition, Springer

[3] Kalman, R. E.; “A New Approach to Linear Filtering and Prediction Problems, Transactions of the ASME - Journal of Basic Engineering” Vol. 82: pag. 35-45 (1960).

[4] Fernandez, B., Alferes, J., Zalama, E., Gómez, J.,”Diseño y simulación de un filtro de Kalman para un robot móvil”.

[5] Kiriy,Evgeni y Buehler, Martin. “Three-state Extended Kalman Filter for Mobile Localization”. Abril 2002.

[6] Kovvali, N.; Banavar, M.; Spanias, A. “An Introduction to Kalman Filtering with MATLAB examples”. Morgan Claypool Publishers, 2013.

[7] Brockwell, P.; Davis, R. “State-Space Models and the Kalman Recursions”. Springer, 1991.

[8] T. Cipra & A. Rubio; “Kalman filter with a non-linear non-Gaussian observation relation”. Springer (1991).

[9] Mohinder, S. Grewal (2001). “Kalman Filtering” . John Wiley & Sons, Inc. ISBN 0-471-26638-8.

[10] Durrant-Whyte, H. “Introduction to Estimation and the Kalman Filter”. University of Sidney, 2001.

[11] Gómez Bruque, J.L. “Filtros de Kalman Extendido en sistemas estocásticos no lineales con observaciones inciertas”. Universidad de Granada. Trabajo Fin de Máster, 2011.

[12] Castañeda, J.A.; Nieto, M.A.; Ortiz, V.A.” Análisis y aplicación del Filtro de Kalman a una señal con ruido aleatorio” Universidad Tecnológica de Pereira, 2013.

[13] Práctica final de la asignatura Sistemas de Percepción del Máster Universitario en Ingeniería Insdustrial. Final Practice Augmented Reality using Markers 2D.

[14] Página web Parrot. Especificaciones técnicas. https://www.parrot.com/es/drones/parrot-mambo-fly

[15] *Computer Vision Toolbox*. Documentación y recursos de Matlab&Simulink. https://es.mathworks.com/products/computer-vision.html

[16] Edge Detection. Documentación y recursos de Matlab&Simulink. https://es.mathworks.com/help/images/edge-detection.html?lang=en

[17] Optical Flow Using Color Information. Kelson R. T. Aires, Andre M. Santana, Adelardo A. D. Medeiros (2008).

[18] Open CV Tutorials. https://docs.opencv.org/master/

[19] John Kristoff. “The Trouble with UDP Scanning” DePaul University. Marzo 2002.

[20] Douglas E. Comer, “Internetworking with TCP/IP Vol. 1: Principles, Protocols, and Architecture. 5th Editi ” Pti H ll 20065th Edition”, Prentice Hall 2006.– Capítulo 11: User Datagram Protocol (UDP).

[21] RFC 768 (UDP, User Datagram Protocol)

[22] Gerardus Blokdyk. “Bluetooth Low Energy A Complete Guide”. Marzo 2019

[23] Philip M. Parker Ph.D. “The 2020-2025 World Outlook for Bluetooth Low Energy”. Octubre 2018.

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR

Universidad de Alcalá